

Magma
7.0.1206

Fri Nov 30 04:33:35 2018

Contents

1	Removing Legacy Authentication Support	3
2	Data Structure Index	5
2.1	Data Structures	5
3	File Index	7
3.1	File List	7
4	Data Structure Documentation	17
4.1	__attribute__ Struct Reference	17
4.1.1	Detailed Description	18
4.1.2	Field Documentation	18
4.1.2.1	"@52	18
4.1.2.2	agent	18
4.1.2.3	body	18
4.1.2.4	connection	18
4.1.2.5	cookie	18
4.1.2.6	cookie	18
4.1.2.7	expunge	18
4.1.2.8	headers	18
4.1.2.9	host	18
4.1.2.10	id	18
4.1.2.11	location	18
4.1.2.12	merged	19
4.1.2.13	method	19
4.1.2.14	mode	19
4.1.2.15	pairs	19
4.1.2.16	params	19
4.1.2.17	port	19
4.1.2.18	portal	19
4.1.2.19	request	19

4.1.2.20	response	19
4.1.2.21	session	19
4.1.2.22	session_state	19
4.1.2.23	user	19
4.1.2.24	username	20
4.1.2.25	usernum	20
4.2	api_lookup_t Struct Reference	21
4.2.1	Detailed Description	21
4.2.2	Field Documentation	21
4.2.2.1	callback	21
4.2.2.2	string	21
4.3	auth_legacy_t Struct Reference	22
4.3.1	Detailed Description	22
4.3.2	Field Documentation	22
4.3.2.1	key	22
4.3.2.2	token	22
4.4	auth_stacie_t Struct Reference	23
4.4.1	Detailed Description	23
4.4.2	Field Documentation	23
4.4.2.1	ephemeral	23
4.4.2.2	keys	23
4.4.2.3	master	23
4.4.2.4	password	23
4.4.2.5	tokens	23
4.4.2.6	verification	24
4.5	auth_t Struct Reference	25
4.5.1	Detailed Description	25
4.5.2	Field Documentation	25
4.5.2.1	bonus	25
4.5.2.2	ephemeral	25
4.5.2.3	key	26
4.5.2.4	keys	26
4.5.2.5	legacy	26
4.5.2.6	locked	26
4.5.2.7	master	26
4.5.2.8	nonce	26
4.5.2.9	salt	26
4.5.2.10	seasoning	26
4.5.2.11	status	26

4.5.2.12	token	26
4.5.2.13	tokens	27
4.5.2.14	username	27
4.5.2.15	usernum	27
4.5.2.16	verification	27
4.6	batch_heap_t Struct Reference	28
4.6.1	Detailed Description	28
4.6.2	Field Documentation	28
4.6.2.1	heap	28
4.6.2.2	points	28
4.6.2.3	r	28
4.6.2.4	scalars	28
4.6.2.5	size	28
4.7	cache_keys_t Struct Reference	29
4.7.1	Detailed Description	29
4.7.2	Field Documentation	29
4.7.2.1	description	29
4.7.2.2	name	29
4.7.2.3	norm	29
4.7.2.4	offset	29
4.7.2.5	required	29
4.8	cache_t Struct Reference	30
4.8.1	Detailed Description	30
4.8.2	Field Documentation	30
4.8.2.1	name	30
4.8.2.2	port	30
4.8.2.3	weight	30
4.9	cached_object Struct Reference	31
4.9.1	Detailed Description	31
4.9.2	Field Documentation	31
4.9.2.1	data	31
4.9.2.2	dtype	32
4.9.2.3	expiration	32
4.9.2.4	id	32
4.9.2.5	next	32
4.9.2.6	persists	32
4.9.2.7	prev	32
4.9.2.8	relaxed	32
4.9.2.9	shadow	33

4.9.2.10	timestamp	33
4.9.2.11	ttl	33
4.10	cached_store_t Struct Reference	34
4.10.1	Detailed Description	34
4.10.2	Field Documentation	34
4.10.2.1	clone	34
4.10.2.2	description	34
4.10.2.3	deserialize	35
4.10.2.4	destructor	35
4.10.2.5	dtype	35
4.10.2.6	dump	35
4.10.2.7	head	35
4.10.2.8	internal	35
4.10.2.9	lock	35
4.10.2.10	serialize	35
4.11	derror_t Struct Reference	36
4.11.1	Detailed Description	36
4.11.2	Field Documentation	36
4.11.2.1	code	36
4.11.2.2	message	36
4.12	dime_ctx Struct Reference	37
4.12.1	Detailed Description	37
4.12.2	Field Documentation	37
4.12.2.1	log_callback	37
4.13	dime_record_t Struct Reference	38
4.13.1	Detailed Description	38
4.13.2	Field Documentation	38
4.13.2.1	dx	38
4.13.2.2	expiry	39
4.13.2.3	policy	39
4.13.2.4	pubkey	39
4.13.2.5	subdomain	39
4.13.2.6	syndicates	39
4.13.2.7	tlssig	39
4.13.2.8	validated	39
4.13.2.9	version	40
4.14	dmime_chunk_key_t Struct Reference	41
4.14.1	Detailed Description	41
4.14.2	Field Documentation	41

4.14.2.1	auth_keyslot	41
4.14.2.2	description	41
4.14.2.3	dest_keyslot	41
4.14.2.4	encrypted	41
4.14.2.5	name	41
4.14.2.6	orig_keyslot	41
4.14.2.7	payload	42
4.14.2.8	recp_keyslot	42
4.14.2.9	required	42
4.14.2.10	section	42
4.14.2.11	sequential	42
4.14.2.12	unique	42
4.15	dmime_common_headers_t Struct Reference	43
4.15.1	Detailed Description	43
4.15.2	Field Documentation	43
4.15.2.1	headers	43
4.16	dmime_envelope_object_t Struct Reference	44
4.16.1	Detailed Description	44
4.16.2	Field Documentation	44
4.16.2.1	auth_recip	44
4.16.2.2	auth_recip_fp	44
4.16.2.3	dest_orig	44
4.16.2.4	dest_orig_fp	44
4.17	dmime_header_key_t Struct Reference	45
4.17.1	Detailed Description	45
4.17.2	Field Documentation	45
4.17.2.1	label	45
4.17.2.2	label_length	45
4.17.2.3	required	45
4.18	dmime_message_t Struct Reference	46
4.18.1	Detailed Description	46
4.18.2	Field Documentation	46
4.18.2.1	attach	46
4.18.2.2	author_full_sig	46
4.18.2.3	author_tree_sig	46
4.18.2.4	common_headers	46
4.18.2.5	destination	46
4.18.2.6	dime_num	47
4.18.2.7	display	47

4.18.2.8	ephemeral	47
4.18.2.9	origin	47
4.18.2.10	origin_display_bounce_sig	47
4.18.2.11	origin_full_sig	47
4.18.2.12	origin_meta_bounce_sig	47
4.18.2.13	other_headers	47
4.18.2.14	size	47
4.18.2.15	state	47
4.18.2.16	tracing	47
4.19	dmime_object_t Struct Reference	48
4.19.1	Detailed Description	48
4.19.2	Field Documentation	48
4.19.2.1	actor	48
4.19.2.2	attach	48
4.19.2.3	author	48
4.19.2.4	common_headers	48
4.19.2.5	destination	48
4.19.2.6	display	49
4.19.2.7	fp_author	49
4.19.2.8	fp_destination	49
4.19.2.9	fp_origin	49
4.19.2.10	fp_recipient	49
4.19.2.11	origin	49
4.19.2.12	other_headers	49
4.19.2.13	recipient	49
4.19.2.14	signet_author	49
4.19.2.15	signet_destination	49
4.19.2.16	signet_origin	49
4.19.2.17	signet_recipient	49
4.19.2.18	state	50
4.20	dmtpl_argument_t Struct Reference	51
4.20.1	Detailed Description	51
4.20.2	Field Documentation	51
4.20.2.1	arg_name	51
4.20.2.2	arg_name_len	51
4.20.2.3	size	51
4.20.2.4	type	51
4.21	dmtpl_command_key_t Struct Reference	52
4.21.1	Detailed Description	52

4.21.2	Field Documentation	52
4.21.2.1	args	52
4.21.2.2	com_name	52
4.21.2.3	com_name_len	52
4.22	dmtplib_command_t Struct Reference	53
4.22.1	Detailed Description	53
4.22.2	Field Documentation	53
4.22.2.1	args	53
4.22.2.2	type	53
4.23	dmtplib_session_t Struct Reference	54
4.23.1	Detailed Description	54
4.23.2	Field Documentation	54
4.23.2.1	_fd	54
4.23.2.2	_inbuf	54
4.23.2.3	_inpos	54
4.23.2.4	active	55
4.23.2.5	con	55
4.23.2.6	domain	55
4.23.2.7	drec	55
4.23.2.8	dx	55
4.23.2.9	mode	55
4.24	dnskey Struct Reference	56
4.24.1	Detailed Description	56
4.24.2	Field Documentation	56
4.24.2.1	algorithm	56
4.24.2.2	do_cache	56
4.24.2.3	dse	56
4.24.2.4	is_sep	57
4.24.2.5	is_zone	57
4.24.2.6	keytag	57
4.24.2.7	label	57
4.24.2.8	pubkey	57
4.24.2.9	rdata	57
4.24.2.10	rdlen	57
4.24.2.11	signkeys	57
4.24.2.12	validated	58
4.25	ds Struct Reference	59
4.25.1	Detailed Description	59
4.25.2	Field Documentation	59

4.25.2.1	algorithm	59
4.25.2.2	digest	59
4.25.2.3	digest_type	59
4.25.2.4	diglen	59
4.25.2.5	keytag	60
4.25.2.6	label	60
4.25.2.7	signkeys	60
4.25.2.8	validated	60
4.26	ED25519_KEY Struct Reference	61
4.26.1	Detailed Description	61
4.26.2	Field Documentation	61
4.26.2.1	private_key	61
4.26.2.2	public_key	61
4.27	encrypt_ctx Struct Reference	62
4.27.1	Detailed Description	62
4.27.2	Field Documentation	62
4.27.2.1	ecies_envelope_evp	62
4.27.2.2	encryption_group	62
4.28	encrypt_keypair_t Struct Reference	63
4.28.1	Detailed Description	63
4.28.2	Field Documentation	63
4.28.2.1	unused	63
4.29	err_desc_t Struct Reference	64
4.29.1	Detailed Description	64
4.29.2	Field Documentation	64
4.29.2.1	errcode	64
4.29.2.2	errmsg	64
4.30	ge25519_niels_t Struct Reference	65
4.30.1	Detailed Description	65
4.30.2	Field Documentation	65
4.30.2.1	t2d	65
4.30.2.2	xaddy	65
4.30.2.3	ysubx	65
4.31	ge25519_p1p1_t Struct Reference	66
4.31.1	Detailed Description	66
4.31.2	Field Documentation	66
4.31.2.1	t	66
4.31.2.2	x	66
4.31.2.3	y	66

4.31.2.4	z	66
4.32	ge25519_pniels_t Struct Reference	67
4.32.1	Detailed Description	67
4.32.2	Field Documentation	67
4.32.2.1	t2d	67
4.32.2.2	xaddy	67
4.32.2.3	ysubx	67
4.32.2.4	z	67
4.33	ge25519_t Struct Reference	68
4.33.1	Detailed Description	68
4.33.2	Field Documentation	68
4.33.2.1	t	68
4.33.2.2	x	68
4.33.2.3	y	68
4.33.2.4	z	68
4.34	ge_cached Struct Reference	69
4.34.1	Detailed Description	69
4.34.2	Field Documentation	69
4.34.2.1	T2d	69
4.34.2.2	YminusX	69
4.34.2.3	YplusX	69
4.34.2.4	Z	69
4.35	ge_plp1 Struct Reference	70
4.35.1	Detailed Description	70
4.35.2	Field Documentation	70
4.35.2.1	T	70
4.35.2.2	X	70
4.35.2.3	Y	70
4.35.2.4	Z	70
4.36	ge_p2 Struct Reference	71
4.36.1	Detailed Description	71
4.36.2	Field Documentation	71
4.36.2.1	X	71
4.36.2.2	Y	71
4.36.2.3	Z	71
4.37	ge_p3 Struct Reference	72
4.37.1	Detailed Description	72
4.37.2	Field Documentation	72
4.37.2.1	T	72

4.37.2.2	X	72
4.37.2.3	Y	72
4.37.2.4	Z	72
4.38	ge_precomp Struct Reference	73
4.38.1	Detailed Description	73
4.38.2	Field Documentation	73
4.38.2.1	xy2d	73
4.38.2.2	yminusx	73
4.38.2.3	yplusx	73
4.39	generated_data_t Struct Reference	74
4.39.1	Detailed Description	74
4.39.2	Field Documentation	74
4.39.2.1	pk	74
4.39.2.2	sig	74
4.39.2.3	valid	74
4.40	http_content_t Struct Reference	75
4.40.1	Detailed Description	75
4.40.2	Field Documentation	75
4.40.2.1	location	75
4.40.2.2	next	75
4.40.2.3	resource	75
4.40.2.4	type	75
4.41	http_data_t Struct Reference	76
4.41.1	Detailed Description	76
4.41.2	Field Documentation	76
4.41.2.1	name	76
4.41.2.2	source	76
4.41.2.3	value	76
4.42	http_page_t Struct Reference	77
4.42.1	Detailed Description	77
4.42.2	Field Documentation	77
4.42.2.1	content	77
4.42.2.2	doc_ctx	77
4.42.2.3	doc_obj	77
4.42.2.4	xpath_ctx	77
4.43	imap_fetch_dataitems_t Struct Reference	78
4.43.1	Detailed Description	78
4.43.2	Field Documentation	78
4.43.2.1	body	78

4.43.2.2	bodystructure	78
4.43.2.3	envelope	78
4.43.2.4	flags	78
4.43.2.5	internaldate	79
4.43.2.6	normal	79
4.43.2.7	normal_partial	79
4.43.2.8	peek	79
4.43.2.9	peek_partial	79
4.43.2.10	rfc822	79
4.43.2.11	rfc822_header	79
4.43.2.12	rfc822_size	79
4.43.2.13	rfc822_text	79
4.43.2.14	uid	80
4.44	imap_fetch_response_t Struct Reference	81
4.44.1	Detailed Description	81
4.44.2	Field Documentation	81
4.44.2.1	key	81
4.44.2.2	next	81
4.44.2.3	value	81
4.45	imap_folder_status_t Struct Reference	82
4.45.1	Detailed Description	82
4.45.2	Field Documentation	82
4.45.2.1	first	82
4.45.2.2	foldernum	82
4.45.2.3	messages	82
4.45.2.4	recent	82
4.45.2.5	uidnext	82
4.45.2.6	unseen	82
4.46	ip_t Struct Reference	83
4.46.1	Detailed Description	83
4.46.2	Field Documentation	83
4.46.2.1	"@6	83
4.46.2.2	family	83
4.46.2.3	ip	83
4.46.2.4	ip4	83
4.46.2.5	ip6	83
4.47	key_pair_t Struct Reference	84
4.47.1	Detailed Description	84
4.47.2	Field Documentation	84

4.47.2.1	private	84
4.47.2.2	public	84
4.48	log_level_t Struct Reference	85
4.48.1	Detailed Description	85
4.48.2	Field Documentation	85
4.48.2.1	code	85
4.48.2.2	name	85
4.49	magma_keys_t Struct Reference	86
4.49.1	Detailed Description	86
4.49.2	Field Documentation	86
4.49.2.1	database	86
4.49.2.2	description	86
4.49.2.3	file	86
4.49.2.4	name	86
4.49.2.5	norm	86
4.49.2.6	overwrite	87
4.49.2.7	required	87
4.49.2.8	set	87
4.49.2.9	store	87
4.50	magma_t Struct Reference	88
4.50.1	Detailed Description	91
4.50.2	Field Documentation	91
4.50.2.1	abuse	91
4.50.2.2	active	91
4.50.2.3	address_length_limit	91
4.50.2.4	admin	91
4.50.2.5	allow_cross_domain	91
4.50.2.6	available	91
4.50.2.7	blacklists	91
4.50.2.8	bypass_addr	92
4.50.2.9	bypass_subnets	92
4.50.2.10	cache	92
4.50.2.11	cache	92
4.50.2.12	close	92
4.50.2.13	config	92
4.50.2.14	connections	92
4.50.2.15	contact	92
4.50.2.16	contact	92
4.50.2.17	content	92

4.50.2.18 core_dump_size_limit	92
4.50.2.19 count	93
4.50.2.20 count	93
4.50.2.21 cryptography	93
4.50.2.22 daemonize	93
4.50.2.23 database	93
4.50.2.24 dhparams_large_keys	93
4.50.2.25 dhparams_rotate	93
4.50.2.26 dime	93
4.50.2.27 dkim	93
4.50.2.28 domain	93
4.50.2.29 domain	93
4.50.2.30 enable	94
4.50.2.31 enable_core_dumps	94
4.50.2.32 enabled	94
4.50.2.33 file	94
4.50.2.34 file	94
4.50.2.35 file	94
4.50.2.36 fonts	94
4.50.2.37 function	94
4.50.2.38 helo_length_limit	94
4.50.2.39 host	94
4.50.2.40 host	95
4.50.2.41 host	95
4.50.2.42 host	95
4.50.2.43 http	95
4.50.2.44 http	95
4.50.2.45 iface	95
4.50.2.46 imap	95
4.50.2.47 impersonate_user	95
4.50.2.48 increase_resource_limits	95
4.50.2.49 indent	95
4.50.2.50 init	96
4.50.2.51 key	96
4.50.2.52 length	96
4.50.2.53 library	96
4.50.2.54 line	96
4.50.2.55 links	96
4.50.2.56 location	96

4.50.2.57 lock	96
4.50.2.58 log	96
4.50.2.59 memory	96
4.50.2.60 message_length_limit	96
4.50.2.61 minimum_password_length	97
4.50.2.62 name	97
4.50.2.63 network_buffer	97
4.50.2.64 number	97
4.50.2.65 output	97
4.50.2.66 output_config	97
4.50.2.67 output_resource_limits	97
4.50.2.68 page_length	97
4.50.2.69 pages	97
4.50.2.70 password	97
4.50.2.71 path	98
4.50.2.72 pool	98
4.50.2.73 pool	98
4.50.2.74 pool	98
4.50.2.75 port	98
4.50.2.76 portal	98
4.50.2.77 premium	98
4.50.2.78 recipient_limit	98
4.50.2.79 registration	98
4.50.2.80 relay	98
4.50.2.81 relay_limit	98
4.50.2.82 retry	98
4.50.2.83 root	99
4.50.2.84 root_directory	99
4.50.2.85 safeguard	99
4.50.2.86 salt	99
4.50.2.87 schema	99
4.50.2.88 secure	99
4.50.2.89 seed_length	99
4.50.2.90 selector	99
4.50.2.91 sender	99
4.50.2.92 servers	99
4.50.2.93 session_timeout	100
4.50.2.94 sessions	100
4.50.2.95 signatures	100

4.50.2.96	signet	100
4.50.2.97	smtp	100
4.50.2.98	socket_path	100
4.50.2.99	spf	100
4.50.2.100	spool	100
4.50.2.101	stack	100
4.50.2.102	standard	100
4.50.2.103	statistics	101
4.50.2.104	storage	101
4.50.2.105	system	101
4.50.2.106	tank	101
4.50.2.107	templates	101
4.50.2.108	thead_stack_size	101
4.50.2.109	time	101
4.50.2.110	timeout	101
4.50.2.111	tls_redirect	101
4.50.2.112	unload	102
4.50.2.113	user	102
4.50.2.114	virus	102
4.50.2.115	web	102
4.50.2.116	worker_threads	102
4.50.2.117	wrap_line_length	102
4.51	mail_cache_t Struct Reference	103
4.51.1	Detailed Description	103
4.51.2	Field Documentation	103
4.51.2.1	messagenum	103
4.51.2.2	text	103
4.52	mail_message_t Struct Reference	104
4.52.1	Detailed Description	104
4.52.2	Field Documentation	104
4.52.2.1	date	104
4.52.2.2	from	104
4.52.2.3	header_length	104
4.52.2.4	mime	104
4.52.2.5	subject	104
4.52.2.6	text	105
4.52.2.7	to	105
4.53	mail_mime_t Struct Reference	106
4.53.1	Detailed Description	106

4.53.2	Field Documentation	106
4.53.2.1	body	106
4.53.2.2	boundary	106
4.53.2.3	children	106
4.53.2.4	encoding	106
4.53.2.5	entire	106
4.53.2.6	header	107
4.53.2.7	type	107
4.54	mappings_t Struct Reference	108
4.54.1	Detailed Description	108
4.54.2	Field Documentation	108
4.54.2.1	base64	108
4.54.2.2	base64_mod	108
4.54.2.3	characters	108
4.54.2.4	values	108
4.54.2.5	zbase32	108
4.55	media_type_t Struct Reference	109
4.55.1	Detailed Description	109
4.55.2	Field Documentation	109
4.55.2.1	bin	109
4.55.2.2	extension	109
4.55.2.3	name	109
4.56	meta_stats_tag_t Struct Reference	110
4.56.1	Detailed Description	110
4.56.2	Field Documentation	110
4.56.2.1	count	110
4.56.2.2	tag	110
4.57	multi_t Struct Reference	111
4.57.1	Detailed Description	111
4.57.2	Field Documentation	111
4.57.2.1	binary	111
4.57.2.2	bl	111
4.57.2.3	dbl	111
4.57.2.4	fl	111
4.57.2.5	i16	112
4.57.2.6	i32	112
4.57.2.7	i64	112
4.57.2.8	i8	112
4.57.2.9	ns	112

4.57.2.10	st	112
4.57.2.11	type	112
4.57.2.12	u16	113
4.57.2.13	u32	113
4.57.2.14	u64	113
4.57.2.15	u8	113
4.57.2.16	val	113
4.58	mx_record_t Struct Reference	114
4.58.1	Detailed Description	114
4.58.2	Field Documentation	114
4.58.2.1	name	114
4.58.2.2	pref	114
4.59	nvp_t Struct Reference	115
4.59.1	Detailed Description	115
4.59.2	Member Function Documentation	115
4.59.2.1	__attribute__	115
4.59.3	Field Documentation	115
4.59.3.1	options	115
4.59.3.2	pairs	115
4.59.3.3	tokens	115
4.60	object_cache_t Struct Reference	116
4.60.1	Detailed Description	116
4.60.2	Field Documentation	116
4.60.2.1	meta	116
4.60.2.2	sessions	116
4.61	object_chunk Struct Reference	117
4.61.1	Detailed Description	117
4.61.2	Field Documentation	117
4.61.2.1	data	117
4.61.2.2	data_size	117
4.61.2.3	flags	117
4.61.2.4	next	117
4.61.2.5	type	117
4.62	packedelem32_t Union Reference	118
4.62.1	Detailed Description	118
4.62.2	Field Documentation	118
4.62.2.1	u	118
4.62.2.2	v	118
4.63	packedelem64_t Union Reference	119

4.63.1	Detailed Description	119
4.63.2	Field Documentation	119
4.63.2.1	u	119
4.63.2.2	v	119
4.64	packedelem8_t Union Reference	120
4.64.1	Detailed Description	120
4.64.2	Field Documentation	120
4.64.2.1	u	120
4.64.2.2	v	120
4.65	queue_t Struct Reference	121
4.65.1	Detailed Description	121
4.65.2	Member Function Documentation	121
4.65.2.1	requeue	121
4.65.3	Field Documentation	121
4.65.3.1	data	121
4.65.3.2	function	121
4.65.3.3	next	121
4.66	random_data_t Struct Reference	122
4.66.1	Detailed Description	122
4.66.2	Field Documentation	122
4.66.2.1	m	122
4.66.2.2	sk	122
4.67	register_session_t Struct Reference	123
4.67.1	Detailed Description	123
4.67.2	Field Documentation	123
4.67.2.1	hvf_input	123
4.67.2.2	hvf_value	123
4.67.2.3	name	123
4.67.2.4	password	123
4.67.2.5	plan	123
4.67.2.6	username	124
4.67.2.7	usernum	124
4.68	relay_keys_t Struct Reference	125
4.68.1	Detailed Description	125
4.68.2	Field Documentation	125
4.68.2.1	description	125
4.68.2.2	name	125
4.68.2.3	norm	125
4.68.2.4	offset	125

4.68.2.5	required	125
4.69	relay_t Struct Reference	126
4.69.1	Detailed Description	126
4.69.2	Field Documentation	126
4.69.2.1	name	126
4.69.2.2	port	126
4.69.2.3	premium	126
4.69.2.4	secure	126
4.70	server_config_t Struct Reference	127
4.70.1	Detailed Description	127
4.70.2	Field Documentation	127
4.70.2.1	bad_command_cutoff	127
4.70.2.2	bad_command_delay	127
4.70.2.3	banner	127
4.70.2.4	certificate	127
4.70.2.5	domain	127
4.70.2.6	listen_queue	128
4.70.2.7	name	128
4.70.2.8	next	128
4.70.2.9	normal_sd	128
4.70.2.10	socket_timeout	128
4.70.2.11	spam	128
4.70.2.12	tcp_port	128
4.70.2.13	tls_ctx	128
4.70.2.14	tls_port	128
4.70.2.15	tls_sd	128
4.71	server_keys_t Struct Reference	129
4.71.1	Detailed Description	129
4.71.2	Field Documentation	129
4.71.2.1	description	129
4.71.2.2	name	129
4.71.2.3	norm	129
4.71.2.4	offset	129
4.71.2.5	required	129
4.72	server_t Struct Reference	130
4.72.1	Detailed Description	130
4.72.2	Field Documentation	130
4.72.2.1	certificate	130
4.72.2.2	context	130

4.72.2.3	cutoff	130
4.72.2.4	delay	131
4.72.2.5	domain	131
4.72.2.6	enabled	131
4.72.2.7	ipv6	131
4.72.2.8	listen_queue	131
4.72.2.9	name	131
4.72.2.10	network	131
4.72.2.11	port	131
4.72.2.12	protocol	131
4.72.2.13	sockd	132
4.72.2.14	timeout	132
4.72.2.15	tls	132
4.72.2.16	type	132
4.72.2.17	violations	132
4.73	sha_databuf_t Struct Reference	133
4.73.1	Detailed Description	133
4.73.2	Field Documentation	133
4.73.2.1	data	133
4.73.2.2	len	133
4.74	signet_field_key_t Struct Reference	134
4.74.1	Detailed Description	134
4.74.2	Field Documentation	134
4.74.2.1	bytes_data_size	134
4.74.2.2	bytes_name_size	134
4.74.2.3	data_size	134
4.74.2.4	data_type	134
4.74.2.5	description	134
4.74.2.6	name	135
4.74.2.7	required	135
4.74.2.8	unique	135
4.75	signet_field_t Struct Reference	136
4.75.1	Detailed Description	136
4.75.2	Field Documentation	136
4.75.2.1	data_offset	136
4.75.2.2	data_size	136
4.75.2.3	id_offset	136
4.75.2.4	key	136
4.75.2.5	name_offset	136

4.75.2.6	name_size	136
4.75.2.7	next	136
4.75.2.8	signet	137
4.76	signet_t Struct Reference	138
4.76.1	Detailed Description	138
4.76.2	Field Documentation	138
4.76.2.1	data	138
4.76.2.2	fields	138
4.76.2.3	size	138
4.76.2.4	type	138
4.77	smtp_inbound_filter_t Struct Reference	139
4.77.1	Detailed Description	139
4.77.2	Field Documentation	139
4.77.2.1	action	139
4.77.2.2	expression	139
4.77.2.3	field	139
4.77.2.4	foldernum	139
4.77.2.5	label	139
4.77.2.6	location	140
4.77.2.7	rulenum	140
4.77.2.8	type	140
4.78	smtp_inbound_prefs_t Struct Reference	141
4.78.1	Detailed Description	141
4.78.2	Field Documentation	142
4.78.2.1	address	142
4.78.2.2	autoreply	142
4.78.2.3	bounces	142
4.78.2.4	daily_rcv_limit	142
4.78.2.5	daily_rcv_limit_ip	142
4.78.2.6	dkim	142
4.78.2.7	dkimaction	142
4.78.2.8	domain	142
4.78.2.9	filters	142
4.78.2.10	foldernum	143
4.78.2.11	forwarded	143
4.78.2.12	greylist	143
4.78.2.13	greytime	143
4.78.2.14	inbox	143
4.78.2.15	local_size	143

4.78.2.16	mark	143
4.78.2.17	messagenum	143
4.78.2.18	next	143
4.78.2.19	outcome	144
4.78.2.20	overquota	144
4.78.2.21	phish	144
4.78.2.22	phishaction	144
4.78.2.23	quota	144
4.78.2.24	rbl	144
4.78.2.25	rblaction	144
4.78.2.26	rcptto	144
4.78.2.27	recv_size_limit	144
4.78.2.28	rollout	145
4.78.2.29	secure	145
4.78.2.30	signet	145
4.78.2.31	signum	145
4.78.2.32	spam	145
4.78.2.33	spam_checked	145
4.78.2.34	spamaction	145
4.78.2.35	spamkey	145
4.78.2.36	spamsig	145
4.78.2.37	spf	146
4.78.2.38	spfaction	146
4.78.2.39	stor_size	146
4.78.2.40	usernum	146
4.78.2.41	virus	146
4.78.2.42	virusaction	146
4.79	smtp_message_t Struct Reference	147
4.79.1	Detailed Description	147
4.79.2	Field Documentation	147
4.79.2.1	date	147
4.79.2.2	from	147
4.79.2.3	header_length	147
4.79.2.4	id	147
4.79.2.5	subject	147
4.79.2.6	text	148
4.79.2.7	to	148
4.80	smtp_outbound_prefs_t Struct Reference	149
4.80.1	Detailed Description	149

4.80.2	Field Documentation	149
4.80.2.1	daily_send_limit	149
4.80.2.2	domain	149
4.80.2.3	importance	149
4.80.2.4	recipients	149
4.80.2.5	send_size_limit	149
4.80.2.6	sent_today	150
4.80.2.7	tls	150
4.80.2.8	usernum	150
4.81	smtp_recipients_t Struct Reference	151
4.81.1	Detailed Description	151
4.81.2	Field Documentation	151
4.81.2.1	address	151
4.81.2.2	next	151
4.82	smtp_session_t Struct Reference	152
4.82.1	Detailed Description	152
4.82.2	Field Documentation	152
4.82.2.1	authenticated	152
4.82.2.2	bypass	152
4.82.2.3	checked	152
4.82.2.4	dkim	152
4.82.2.5	esmtplib	152
4.82.2.6	helo	153
4.82.2.7	in_prefs	153
4.82.2.8	ip_address_v4	153
4.82.2.9	mailfrom	153
4.82.2.10	max_length	153
4.82.2.11	message	153
4.82.2.12	num_recipients	153
4.82.2.13	out_prefs	153
4.82.2.14	rbl	153
4.82.2.15	spf	153
4.82.2.16	submission	153
4.82.2.17	suggested_eight_bit	153
4.82.2.18	suggested_length	154
4.82.2.19	virus	154
4.83	statistics_vp_t Struct Reference	155
4.83.1	Detailed Description	155
4.83.2	Field Documentation	155

4.83.2.1	stmt	155
4.83.2.2	val	155
4.84	subnet_t Struct Reference	156
4.84.1	Detailed Description	156
4.84.2	Field Documentation	156
4.84.2.1	address	156
4.84.2.2	mask	156
4.85	teacher_data_t Struct Reference	157
4.85.1	Detailed Description	157
4.85.2	Field Documentation	157
4.85.2.1	completed	157
4.85.2.2	disposition	157
4.85.2.3	keynum	157
4.85.2.4	password	157
4.85.2.5	signature	157
4.85.2.6	signum	158
4.85.2.7	username	158
4.85.2.8	usernum	158
4.86	test_data_t Struct Reference	159
4.86.1	Detailed Description	159
4.86.2	Field Documentation	159
4.86.2.1	m	159
4.86.2.2	pk	159
4.86.2.3	sig	159
4.86.2.4	sk	159
4.87	tok_state_t Struct Reference	160
4.87.1	Detailed Description	160
4.87.2	Field Documentation	160
4.87.2.1	block	160
4.87.2.2	length	160
4.87.2.3	position	160
4.87.2.4	remaining	160
4.87.2.5	token	160
5	File Documentation	161
5.1	src/core/buckets/arrays.c File Reference	161
5.1.1	Function Documentation	162
5.1.1.1	ar_alloc	162
5.1.1.2	ar_append	162

5.1.1.3	ar_avail_get	162
5.1.1.4	ar_dupe	162
5.1.1.5	ar_field_ar	163
5.1.1.6	ar_field_ns	163
5.1.1.7	ar_field_pl	163
5.1.1.8	ar_field_ptr	164
5.1.1.9	ar_field_st	164
5.1.1.10	ar_field_type	164
5.1.1.11	ar_free	164
5.1.1.12	ar_length_get	165
5.1.1.13	ar_length_set	165
5.2	src/core/buckets/buckets.h File Reference	166
5.2.1	Define Documentation	168
5.2.1.1	ARRAY_MAX_ELEMENTS	168
5.2.1.2	ARRAY_TYPE_ARRAY	168
5.2.1.3	ARRAY_TYPE_EMPTY	168
5.2.1.4	ARRAY_TYPE_NULLER	168
5.2.1.5	ARRAY_TYPE_PLACER	168
5.2.1.6	ARRAY_TYPE_POINTER	168
5.2.1.7	ARRAY_TYPE_SIZER	169
5.2.1.8	ARRAY_TYPE_STRINGER	169
5.2.1.9	MAGMA_CORE_POOL_OBJECTS_LIMIT	169
5.2.1.10	MAGMA_CORE_POOL_TIMEOUT_LIMIT	169
5.2.2	Typedef Documentation	169
5.2.2.1	array_t	169
5.2.2.2	status_t	169
5.2.3	Enumeration Type Documentation	169
5.2.3.1	M_POOL_STATUS	169
5.2.4	Function Documentation	169
5.2.4.1	__attribute__	169
5.2.4.2	ar_alloc	170
5.2.4.3	ar_append	170
5.2.4.4	ar_avail_get	170
5.2.4.5	ar_dupe	171
5.2.4.6	ar_field_ar	171
5.2.4.7	ar_field_ns	171
5.2.4.8	ar_field_pl	172
5.2.4.9	ar_field_ptr	172
5.2.4.10	ar_field_st	172

5.2.4.11	ar_field_type	172
5.2.4.12	ar_free	173
5.2.4.13	ar_length_get	173
5.2.4.14	ar_length_set	173
5.2.4.15	pool_alloc	174
5.2.4.16	pool_free	174
5.2.4.17	pool_get_available	174
5.2.4.18	pool_get_count	175
5.2.4.19	pool_get_failures	175
5.2.4.20	pool_get_obj	175
5.2.4.21	pool_get_status	176
5.2.4.22	pool_get_timeout	176
5.2.4.23	pool_pull	176
5.2.4.24	pool_release	177
5.2.4.25	pool_set_obj	177
5.2.4.26	pool_set_status	177
5.2.4.27	pool_swap_obj	178
5.2.4.28	stacker_alloc	178
5.2.4.29	stacker_free	178
5.2.4.30	stacker_nodes	179
5.2.4.31	stacker_pop	179
5.2.4.32	stacker_push	179
5.2.5	Variable Documentation	179
5.2.5.1	pool_t	179
5.2.5.2	stacker_node_t	180
5.2.5.3	stacker_t	180
5.3	src/core/buckets/pool.c File Reference	181
5.3.1	Function Documentation	181
5.3.1.1	pool_alloc	181
5.3.1.2	pool_free	182
5.3.1.3	pool_get_available	182
5.3.1.4	pool_get_count	182
5.3.1.5	pool_get_failures	183
5.3.1.6	pool_get_obj	183
5.3.1.7	pool_get_status	183
5.3.1.8	pool_get_timeout	184
5.3.1.9	pool_pull	184
5.3.1.10	pool_release	185
5.3.1.11	pool_set_obj	185

5.3.1.12	pool_set_status	185
5.3.1.13	pool_swap_obj	186
5.4	src/core/buckets/stacked.c File Reference	187
5.4.1	Function Documentation	187
5.4.1.1	stacker_alloc	187
5.4.1.2	stacker_free	187
5.4.1.3	stacker_nodes	188
5.4.1.4	stacker_pop	188
5.4.1.5	stacker_push	188
5.5	src/core/checksum/adler.c File Reference	189
5.5.1	Function Documentation	189
5.5.1.1	hash_adler32	189
5.6	src/core/checksum/checksum.h File Reference	190
5.6.1	Function Documentation	190
5.6.1.1	crc24_checksum	190
5.6.1.2	crc24_final	191
5.6.1.3	crc24_init	191
5.6.1.4	crc24_update	191
5.6.1.5	crc32_checksum	191
5.6.1.6	crc32_update	191
5.6.1.7	crc64_checksum	192
5.6.1.8	crc64_update	192
5.6.1.9	hash_adler32	192
5.6.1.10	hash_fletcher32	193
5.6.1.11	hash_murmur32	193
5.6.1.12	hash_murmur64	193
5.7	src/core/checksum/crc.c File Reference	194
5.7.1	Define Documentation	194
5.7.1.1	A	194
5.7.1.2	A1	195
5.7.1.3	B	195
5.7.1.4	C	195
5.7.1.5	CRC24_INIT	195
5.7.1.6	CRC24_POLY	195
5.7.1.7	D	195
5.7.1.8	S32	195
5.7.1.9	S8	195
5.7.2	Function Documentation	195
5.7.2.1	crc24_checksum	195

5.7.2.2	crc24_final	196
5.7.2.3	crc24_init	196
5.7.2.4	crc24_update	196
5.7.2.5	crc32_checksum	196
5.7.2.6	crc32_update	197
5.7.2.7	crc64_checksum	197
5.7.2.8	crc64_update	197
5.7.3	Variable Documentation	198
5.7.3.1	crc24_table	198
5.7.3.2	crc32_table	198
5.7.3.3	crc64_table	198
5.8	src/core/checksum/fletcher.c File Reference	199
5.8.1	Function Documentation	199
5.8.1.1	hash_fletcher32	199
5.9	src/core/checksum/murmur.c File Reference	200
5.9.1	Function Documentation	200
5.9.1.1	hash_murmur32	200
5.9.1.2	hash_murmur64	200
5.10	src/core/classify/ascii.c File Reference	201
5.10.1	Function Documentation	201
5.10.1.1	chr_alphanumeric	201
5.10.1.2	chr_ascii	202
5.10.1.3	chr_blank	202
5.10.1.4	chr_is_class	202
5.10.1.5	chr_lower	202
5.10.1.6	chr_numeric	203
5.10.1.7	chr_printable	203
5.10.1.8	chr_punctuation	203
5.10.1.9	chr_upper	203
5.10.1.10	chr_whitespace	204
5.11	src/core/classify/classify.h File Reference	205
5.11.1	Function Documentation	205
5.11.1.1	chr_alphanumeric	205
5.11.1.2	chr_ascii	206
5.11.1.3	chr_blank	206
5.11.1.4	chr_is_class	206
5.11.1.5	chr_lower	206
5.11.1.6	chr_numeric	207
5.11.1.7	chr_printable	207

5.11.1.8	chr_punctuation	207
5.11.1.9	chr_upper	207
5.11.1.10	chr_whitespace	208
5.12	src/core/compare/compare.h File Reference	209
5.12.1	Function Documentation	209
5.12.1.1	mm_cmp_ci_eq	209
5.12.1.2	mm_cmp_cs_eq	210
5.12.1.3	st_cmp_ci_ends	210
5.12.1.4	st_cmp_ci_eq	210
5.12.1.5	st_cmp_ci_starts	211
5.12.1.6	st_cmp_cs_ends	211
5.12.1.7	st_cmp_cs_eq	212
5.12.1.8	st_cmp_cs_starts	212
5.12.1.9	st_search_chr	212
5.12.1.10	st_search_ci	213
5.12.1.11	st_search_cs	213
5.13	src/core/compare/ends.c File Reference	214
5.13.1	Function Documentation	214
5.13.1.1	st_cmp_ci_ends	214
5.13.1.2	st_cmp_cs_ends	214
5.14	src/core/compare/equal.c File Reference	215
5.14.1	Function Documentation	215
5.14.1.1	mm_cmp_ci_eq	215
5.14.1.2	mm_cmp_cs_eq	215
5.14.1.3	st_cmp_ci_eq	216
5.14.1.4	st_cmp_cs_eq	216
5.15	src/core/compare/search.c File Reference	217
5.15.1	Function Documentation	217
5.15.1.1	st_search_chr	217
5.15.1.2	st_search_ci	217
5.15.1.3	st_search_cs	218
5.16	src/servers/imap/search.c File Reference	219
5.16.1	Function Documentation	219
5.16.1.1	imap_search_flag	219
5.16.1.2	imap_search_messages	219
5.16.1.3	imap_search_messages_body	219
5.16.1.4	imap_search_messages_date	220
5.16.1.5	imap_search_messages_date_compare	220
5.16.1.6	imap_search_messages_header	220

5.16.1.7	imap_search_messages_inner	220
5.16.1.8	imap_search_messages_range	220
5.16.1.9	imap_search_messages_size	220
5.16.1.10	imap_search_messages_text	221
5.16.2	Variable Documentation	221
5.16.2.1	MONTH_LOOKUP	221
5.17	src/core/compare/starts.c File Reference	222
5.17.1	Function Documentation	222
5.17.1.1	st_cmp_ci_starts	222
5.17.1.2	st_cmp_cs_starts	222
5.18	src/core/core.h File Reference	223
5.18.1	Define Documentation	225
5.18.1.1	__bool_t_defined	225
5.18.1.2	INT24_MAX	225
5.18.1.3	INT24_MIN	225
5.18.1.4	log_check	225
5.18.1.5	log_critical	225
5.18.1.6	log_error	225
5.18.1.7	log_info	226
5.18.1.8	log_options	226
5.18.1.9	log_pedantic	226
5.18.1.10	SIGUNUSED	228
5.18.1.11	UINT24_MAX	228
5.18.1.12	UINT24_MIN	229
5.18.2	Typedef Documentation	229
5.18.2.1	bool_t	229
5.18.2.2	byte_t	229
5.18.2.3	chr_t	229
5.18.2.4	int24_t	229
5.18.2.5	int_t	229
5.18.2.6	uchr_t	229
5.18.2.7	uint24_t	229
5.18.2.8	uint_t	229
5.18.3	Enumeration Type Documentation	229
5.18.3.1	"@0	229
5.18.3.2	M_TYPE	230
5.18.4	Function Documentation	230
5.18.4.1	__attribute__	230
5.18.4.2	type	230

5.18.5	Variable Documentation	231
5.18.5.1	<code>__int24_t</code>	231
5.18.5.2	<code>__uint24_t</code>	231
5.18.5.3	<code>log_enabled</code>	231
5.18.5.4	<code>log_mutex</code>	231
5.19	<code>src/core/encodings/base64.c</code> File Reference	232
5.19.1	Function Documentation	232
5.19.1.1	<code>base64_decode</code>	232
5.19.1.2	<code>base64_decode_mod</code>	233
5.19.1.3	<code>base64_decode_opts</code>	233
5.19.1.4	<code>base64_decoded_length</code>	233
5.19.1.5	<code>base64_decoded_length_mod</code>	234
5.19.1.6	<code>base64_encode</code>	234
5.19.1.7	<code>base64_encode_mod</code>	234
5.19.1.8	<code>base64_encode_opts</code>	234
5.19.1.9	<code>base64_encode_wrap</code>	235
5.19.1.10	<code>base64_encoded_length</code>	235
5.19.1.11	<code>base64_encoded_length_mod</code>	235
5.19.1.12	<code>base64_encoded_length_wrap</code>	235
5.20	<code>src/core/encodings/encodings.h</code> File Reference	237
5.20.1	Define Documentation	239
5.20.1.1	<code>BASE64_LINE_WRAP_LENGTH</code>	239
5.20.1.2	<code>QP_LINE_WRAP_LENGTH</code>	239
5.20.1.3	<code>URL_MAX_LENGTH</code>	239
5.20.2	Enumeration Type Documentation	239
5.20.2.1	<code>base64_wrap_t</code>	239
5.20.3	Function Documentation	239
5.20.3.1	<code>base64_decode</code>	239
5.20.3.2	<code>base64_decode_mod</code>	240
5.20.3.3	<code>base64_decode_opts</code>	240
5.20.3.4	<code>base64_decoded_length</code>	240
5.20.3.5	<code>base64_decoded_length_mod</code>	240
5.20.3.6	<code>base64_encode</code>	241
5.20.3.7	<code>base64_encode_mod</code>	241
5.20.3.8	<code>base64_encode_opts</code>	241
5.20.3.9	<code>base64_encode_wrap</code>	242
5.20.3.10	<code>base64_encoded_length</code>	242
5.20.3.11	<code>base64_encoded_length_mod</code>	242
5.20.3.12	<code>base64_encoded_length_wrap</code>	242

5.20.3.13	hex_count_st	243
5.20.3.14	hex_decode_chr	243
5.20.3.15	hex_decode_opts	243
5.20.3.16	hex_decode_st	244
5.20.3.17	hex_encode_chr	244
5.20.3.18	hex_encode_opts	244
5.20.3.19	hex_encode_st	245
5.20.3.20	hex_encode_st_debug	245
5.20.3.21	hex_valid_chr	245
5.20.3.22	hex_valid_st	246
5.20.3.23	qp_decode	246
5.20.3.24	qp_encode	246
5.20.3.25	url_decode	246
5.20.3.26	url_encode	247
5.20.3.27	url_valid_chr	247
5.20.3.28	url_valid_st	247
5.20.3.29	zbase32_decode	248
5.20.3.30	zbase32_encode	248
5.20.4	Variable Documentation	248
5.20.4.1	mappings	248
5.21	src/core/encodings/hex.c File Reference	249
5.21.1	Function Documentation	249
5.21.1.1	hex_count_st	249
5.21.1.2	hex_decode_chr	250
5.21.1.3	hex_decode_opts	250
5.21.1.4	hex_decode_st	250
5.21.1.5	hex_encode_chr	251
5.21.1.6	hex_encode_opts	251
5.21.1.7	hex_encode_st	251
5.21.1.8	hex_encode_st_debug	252
5.21.1.9	hex_valid_chr	252
5.21.1.10	hex_valid_st	252
5.22	src/core/encodings/mappings.c File Reference	253
5.22.1	Variable Documentation	253
5.22.1.1	mappings	253
5.23	src/core/encodings/qp.c File Reference	254
5.23.1	Function Documentation	254
5.23.1.1	qp_decode	254
5.23.1.2	qp_encode	254

5.24	src/core/encodings/url.c File Reference	255
5.24.1	Function Documentation	255
5.24.1.1	url_decode	255
5.24.1.2	url_encode	255
5.24.1.3	url_valid_chr	256
5.24.1.4	url_valid_st	256
5.25	src/core/encodings/zbase32.c File Reference	257
5.25.1	Function Documentation	257
5.25.1.1	zbase32_decode	257
5.25.1.2	zbase32_encode	257
5.26	src/core/host/backtrace.c File Reference	258
5.26.1	Detailed Description	258
5.26.2	Function Documentation	258
5.26.2.1	backtrace_print	258
5.27	src/core/host/color.c File Reference	259
5.27.1	Function Documentation	259
5.27.1.1	color_blue	259
5.27.1.2	color_blue_bold	260
5.27.1.3	color_blue_intense	260
5.27.1.4	color_blue_intense_bold	260
5.27.1.5	color_blue_underline	260
5.27.1.6	color_cyan	260
5.27.1.7	color_cyan_bold	260
5.27.1.8	color_cyan_intense	260
5.27.1.9	color_cyan_intense_bold	260
5.27.1.10	color_cyan_underline	260
5.27.1.11	color_green	261
5.27.1.12	color_green_bold	261
5.27.1.13	color_green_intense	261
5.27.1.14	color_green_intense_bold	261
5.27.1.15	color_green_underline	261
5.27.1.16	color_purple	261
5.27.1.17	color_purple_bold	261
5.27.1.18	color_purple_intense	261
5.27.1.19	color_purple_intense_bold	261
5.27.1.20	color_purple_underline	262
5.27.1.21	color_red	262
5.27.1.22	color_red_bold	262
5.27.1.23	color_red_intense	262

5.27.1.24	color_red_intense_bold	262
5.27.1.25	color_red_underline	262
5.27.1.26	color_reset	262
5.27.1.27	color_supported	262
5.27.1.28	color_white	263
5.27.1.29	color_white_bold	263
5.27.1.30	color_white_intense	263
5.27.1.31	color_white_intense_bold	263
5.27.1.32	color_white_underline	263
5.27.1.33	color_yellow	263
5.27.1.34	color_yellow_bold	263
5.27.1.35	color_yellow_intense	263
5.27.1.36	color_yellow_intense_bold	263
5.27.1.37	color_yellow_underline	264
5.28	src/core/host/errors.c File Reference	265
5.28.1	Detailed Description	265
5.28.2	Define Documentation	266
5.28.2.1	MAGMA_E2BIG	266
5.28.2.2	MAGMA_EACCES	266
5.28.2.3	MAGMA_EAGAIN	266
5.28.2.4	MAGMA_EBADF	266
5.28.2.5	MAGMA_EBUSY	266
5.28.2.6	MAGMA_ECHILD	266
5.28.2.7	MAGMA_EDOM	266
5.28.2.8	MAGMA_EEXIST	266
5.28.2.9	MAGMA_EFAULT	266
5.28.2.10	MAGMA_EFBIG	267
5.28.2.11	MAGMA_EINTR	267
5.28.2.12	MAGMA_EINVAL	267
5.28.2.13	MAGMA_EIO	267
5.28.2.14	MAGMA_EISDIR	267
5.28.2.15	MAGMA_EMFILE	267
5.28.2.16	MAGMA_EMLINK	267
5.28.2.17	MAGMA_ENFILE	267
5.28.2.18	MAGMA_ENODEV	267
5.28.2.19	MAGMA_ENOENT	268
5.28.2.20	MAGMA_ENOEXEC	268
5.28.2.21	MAGMA_ENOMEM	268
5.28.2.22	MAGMA_ENOSPC	268

5.28.2.23	MAGMA_ENOTBLK	268
5.28.2.24	MAGMA_ENOTDIR	268
5.28.2.25	MAGMA_ENOTTY	268
5.28.2.26	MAGMA_ENXIO	268
5.28.2.27	MAGMA_EPERM	268
5.28.2.28	MAGMA_EPIPE	269
5.28.2.29	MAGMA_ERANGE	269
5.28.2.30	MAGMA_EROFS	269
5.28.2.31	MAGMA_ESPIPE	269
5.28.2.32	MAGMA_ESRCH	269
5.28.2.33	MAGMA_ETXTBSY	269
5.28.2.34	MAGMA_EXDEV	269
5.28.3	Function Documentation	269
5.28.3.1	errno_name	269
5.29	src/servers/http/errors.c File Reference	270
5.29.1	Function Documentation	270
5.29.1.1	http_print_301	270
5.29.1.2	http_print_400	271
5.29.1.3	http_print_403	271
5.29.1.4	http_print_404	271
5.29.1.5	http_print_405	272
5.29.1.6	http_print_500	272
5.29.1.7	http_print_500_log	272
5.29.1.8	http_print_501	272
5.30	src/core/host/files.c File Reference	274
5.30.1	Function Documentation	274
5.30.1.1	file_accessible	274
5.30.1.2	file_load	274
5.30.1.3	file_read	275
5.30.1.4	file_readwritable	275
5.30.1.5	file_temp_handle	275
5.30.1.6	file_world_accessible	276
5.31	src/core/host/folder.c File Reference	277
5.31.1	Function Documentation	277
5.31.1.1	folder_count	277
5.31.1.2	folder_exists	277
5.32	src/core/host/host.c File Reference	278
5.32.1	Function Documentation	278
5.32.1.1	host_platform	278

5.32.1.2	host_version	278
5.33	src/core/host/host.h File Reference	279
5.33.1	Define Documentation	283
5.33.1.1	MAGMA_COLOR_BLUE	283
5.33.1.2	MAGMA_COLOR_BLUE_BOLD	283
5.33.1.3	MAGMA_COLOR_BLUE_INTENSE	283
5.33.1.4	MAGMA_COLOR_BLUE_INTENSE_BOLD	283
5.33.1.5	MAGMA_COLOR_BLUE_UNDERLINE	284
5.33.1.6	MAGMA_COLOR_CYAN	284
5.33.1.7	MAGMA_COLOR_CYAN_BOLD	284
5.33.1.8	MAGMA_COLOR_CYAN_INTENSE	284
5.33.1.9	MAGMA_COLOR_CYAN_INTENSE_BOLD	284
5.33.1.10	MAGMA_COLOR_CYAN_UNDERLINE	284
5.33.1.11	MAGMA_COLOR_GREEN	284
5.33.1.12	MAGMA_COLOR_GREEN_BOLD	284
5.33.1.13	MAGMA_COLOR_GREEN_INTENSE	284
5.33.1.14	MAGMA_COLOR_GREEN_INTENSE_BOLD	285
5.33.1.15	MAGMA_COLOR_GREEN_UNDERLINE	285
5.33.1.16	MAGMA_COLOR_PURPLE	285
5.33.1.17	MAGMA_COLOR_PURPLE_BOLD	285
5.33.1.18	MAGMA_COLOR_PURPLE_INTENSE	285
5.33.1.19	MAGMA_COLOR_PURPLE_INTENSE_BOLD	285
5.33.1.20	MAGMA_COLOR_PURPLE_UNDERLINE	285
5.33.1.21	MAGMA_COLOR_RED	285
5.33.1.22	MAGMA_COLOR_RED_BOLD	285
5.33.1.23	MAGMA_COLOR_RED_INTENSE	286
5.33.1.24	MAGMA_COLOR_RED_INTENSE_BOLD	286
5.33.1.25	MAGMA_COLOR_RED_UNDERLINE	286
5.33.1.26	MAGMA_COLOR_RESET	286
5.33.1.27	MAGMA_COLOR_WHITE	286
5.33.1.28	MAGMA_COLOR_WHITE_BOLD	286
5.33.1.29	MAGMA_COLOR_WHITE_INTENSE	286
5.33.1.30	MAGMA_COLOR_WHITE_INTENSE_BOLD	286
5.33.1.31	MAGMA_COLOR_WHITE_UNDERLINE	286
5.33.1.32	MAGMA_COLOR_YELLOW	287
5.33.1.33	MAGMA_COLOR_YELLOW_BOLD	287
5.33.1.34	MAGMA_COLOR_YELLOW_INTENSE	287
5.33.1.35	MAGMA_COLOR_YELLOW_INTENSE_BOLD	287
5.33.1.36	MAGMA_COLOR_YELLOW_UNDERLINE	287

5.33.1.37	MAGMA_PROC_PATH	287
5.33.2	Typedef Documentation	287
5.33.2.1	octet_t	287
5.33.2.2	segment_t	287
5.33.3	Enumeration Type Documentation	287
5.33.3.1	"@4	287
5.33.4	Function Documentation	288
5.33.4.1	backtrace_print	288
5.33.4.2	color_blue	288
5.33.4.3	color_blue_bold	288
5.33.4.4	color_blue_intense	288
5.33.4.5	color_blue_intense_bold	288
5.33.4.6	color_blue_underline	288
5.33.4.7	color_cyan	288
5.33.4.8	color_cyan_bold	289
5.33.4.9	color_cyan_intense	289
5.33.4.10	color_cyan_intense_bold	289
5.33.4.11	color_cyan_underline	289
5.33.4.12	color_green	289
5.33.4.13	color_green_bold	289
5.33.4.14	color_green_intense	289
5.33.4.15	color_green_intense_bold	289
5.33.4.16	color_green_underline	289
5.33.4.17	color_purple	290
5.33.4.18	color_purple_bold	290
5.33.4.19	color_purple_intense	290
5.33.4.20	color_purple_intense_bold	290
5.33.4.21	color_purple_underline	290
5.33.4.22	color_red	290
5.33.4.23	color_red_bold	290
5.33.4.24	color_red_intense	290
5.33.4.25	color_red_intense_bold	290
5.33.4.26	color_red_underline	291
5.33.4.27	color_reset	291
5.33.4.28	color_supported	291
5.33.4.29	color_white	291
5.33.4.30	color_white_bold	291
5.33.4.31	color_white_intense	291
5.33.4.32	color_white_intense_bold	291

5.33.4.33 color_white_underline	291
5.33.4.34 color_yellow	292
5.33.4.35 color_yellow_bold	292
5.33.4.36 color_yellow_intense	292
5.33.4.37 color_yellow_intense_bold	292
5.33.4.38 color_yellow_underline	292
5.33.4.39 errno_name	292
5.33.4.40 file_accessible	292
5.33.4.41 file_load	293
5.33.4.42 file_read	293
5.33.4.43 file_readwritable	293
5.33.4.44 file_temp_handle	294
5.33.4.45 file_world_accessible	294
5.33.4.46 folder_count	294
5.33.4.47 folder_exists	294
5.33.4.48 host_platform	295
5.33.4.49 host_version	295
5.33.4.50 ip_addr_eq	295
5.33.4.51 ip_addr_st	295
5.33.4.52 ip_copy	296
5.33.4.53 ip_family	296
5.33.4.54 ip_localhost	296
5.33.4.55 ip_matches_subnet	297
5.33.4.56 ip_octet	297
5.33.4.57 ip_presentation	297
5.33.4.58 ip_private	298
5.33.4.59 ip_reversed	298
5.33.4.60 ip_segment	298
5.33.4.61 ip_standard	299
5.33.4.62 ip_subnet	299
5.33.4.63 ip_subnet_st	299
5.33.4.64 ip_type	300
5.33.4.65 ip_word	300
5.33.4.66 process_find_pid	300
5.33.4.67 process_kill	301
5.33.4.68 process_my_pid	301
5.33.4.69 signal_name	301
5.33.4.70 spool_check	302
5.33.4.71 spool_check_file	302

5.33.4.72	spool_cleanup	302
5.33.4.73	spool_error_stats	302
5.33.4.74	spool_mktemp	303
5.33.4.75	spool_path	303
5.33.4.76	spool_start	303
5.33.4.77	spool_stop	304
5.33.4.78	tcp_addr_ip	304
5.33.4.79	tcp_addr_st	304
5.33.4.80	tcp_continue	304
5.33.4.81	tcp_error	304
5.33.4.82	tcp_read	305
5.33.4.83	tcp_status	305
5.33.4.84	tcp_wait	305
5.33.4.85	tcp_write	306
5.34	src/core/host/ip.c File Reference	307
5.34.1	Detailed Description	308
5.34.2	Function Documentation	308
5.34.2.1	ip_addr_eq	308
5.34.2.2	ip_addr_st	308
5.34.2.3	ip_copy	308
5.34.2.4	ip_family	309
5.34.2.5	ip_localhost	309
5.34.2.6	ip_matches_subnet	309
5.34.2.7	ip_octet	310
5.34.2.8	ip_presentation	310
5.34.2.9	ip_private	310
5.34.2.10	ip_reversed	311
5.34.2.11	ip_segment	311
5.34.2.12	ip_standard	311
5.34.2.13	ip_subnet	312
5.34.2.14	ip_subnet_st	312
5.34.2.15	ip_type	312
5.34.2.16	ip_word	313
5.35	src/core/host/process.c File Reference	314
5.35.1	Function Documentation	314
5.35.1.1	process_find_pid	314
5.35.1.2	process_kill	314
5.35.1.3	process_my_pid	315
5.36	src/engine/context/process.c File Reference	316

5.36.1	Function Documentation	316
5.36.1.1	process_maint	316
5.36.1.2	process_start	316
5.36.1.3	process_stop	317
5.36.2	Variable Documentation	317
5.36.2.1	day	317
5.36.2.2	exit_and_dump	317
5.36.2.3	maint	317
5.37	src/core/host/signals.c File Reference	318
5.37.1	Detailed Description	318
5.37.2	Function Documentation	318
5.37.2.1	signal_name	318
5.38	src/core/host/spool.c File Reference	319
5.38.1	Function Documentation	319
5.38.1.1	spool_check	319
5.38.1.2	spool_check_file	320
5.38.1.3	spool_cleanup	320
5.38.1.4	spool_error_stats	320
5.38.1.5	spool_mktemp	320
5.38.1.6	spool_path	321
5.38.1.7	spool_start	321
5.38.1.8	spool_stop	321
5.39	src/core/host/tcp.c File Reference	323
5.39.1	Detailed Description	323
5.39.2	Function Documentation	323
5.39.2.1	tcp_addr_ip	323
5.39.2.2	tcp_addr_st	323
5.39.2.3	tcp_continue	324
5.39.2.4	tcp_error	324
5.39.2.5	tcp_read	324
5.39.2.6	tcp_status	324
5.39.2.7	tcp_wait	325
5.39.2.8	tcp_write	325
5.40	src/core/indexes/cursors.c File Reference	326
5.40.1	Function Documentation	326
5.40.1.1	inx_cursor_alloc	326
5.40.1.2	inx_cursor_free	327
5.40.1.3	inx_cursor_key_active	327
5.40.1.4	inx_cursor_key_next	327

5.40.1.5	<code>inx_cursor_reset</code>	328
5.40.1.6	<code>inx_cursor_value_active</code>	328
5.40.1.7	<code>inx_cursor_value_next</code>	328
5.41	<code>src/core/indexes/hashed.c</code> File Reference	330
5.41.1	Define Documentation	330
5.41.1.1	<code>MAGMA_HASHED_BUCKETS</code>	330
5.41.2	Function Documentation	331
5.41.2.1	<code>__attribute__</code>	331
5.41.2.2	<code>hashed_alloc</code>	331
5.41.2.3	<code>hashed_bucket</code>	331
5.41.2.4	<code>hashed_bucket_add</code>	331
5.41.2.5	<code>hashed_bucket_alloc</code>	332
5.41.2.6	<code>hashed_bucket_find_key</code>	332
5.41.2.7	<code>hashed_bucket_get_ptr</code>	332
5.41.2.8	<code>hashed_bucket_set_ptr</code>	332
5.41.2.9	<code>hashed_cursor_active</code>	332
5.41.2.10	<code>hashed_cursor_alloc</code>	332
5.41.2.11	<code>hashed_cursor_free</code>	333
5.41.2.12	<code>hashed_cursor_key_active</code>	333
5.41.2.13	<code>hashed_cursor_key_next</code>	333
5.41.2.14	<code>hashed_cursor_next</code>	333
5.41.2.15	<code>hashed_cursor_reset</code>	333
5.41.2.16	<code>hashed_cursor_value_active</code>	333
5.41.2.17	<code>hashed_cursor_value_next</code>	333
5.41.2.18	<code>hashed_delete</code>	333
5.41.2.19	<code>hashed_find</code>	334
5.41.2.20	<code>hashed_free</code>	334
5.41.2.21	<code>hashed_insert</code>	334
5.41.2.22	<code>hashed_truncate</code>	334
5.41.3	Variable Documentation	334
5.41.3.1	<code>hashed_bucket_t</code>	334
5.41.3.2	<code>hashed_cursor_t</code>	334
5.41.3.3	<code>hashed_index_t</code>	334
5.42	<code>src/core/indexes/indexes.h</code> File Reference	336
5.42.1	Define Documentation	337
5.42.1.1	<code>MAGMA_INDEX_OPTION</code>	337
5.42.1.2	<code>MAGMA_INDEX_TYPE</code>	337
5.42.2	Enumeration Type Documentation	338
5.42.2.1	<code>MAGMA_INDEX</code>	338

5.42.3	Function Documentation	338
5.42.3.1	__attribute__	338
5.42.3.2	hashed_alloc	338
5.42.3.3	inx_alloc	338
5.42.3.4	inx_append	339
5.42.3.5	inx_auto_read	339
5.42.3.6	inx_auto_unlock	339
5.42.3.7	inx_auto_write	340
5.42.3.8	inx_cleanup	340
5.42.3.9	inx_count	340
5.42.3.10	inx_cursor_alloc	340
5.42.3.11	inx_cursor_free	341
5.42.3.12	inx_cursor_key_active	341
5.42.3.13	inx_cursor_key_next	342
5.42.3.14	inx_cursor_reset	342
5.42.3.15	inx_cursor_value_active	342
5.42.3.16	inx_cursor_value_next	343
5.42.3.17	inx_delete	343
5.42.3.18	inx_find	343
5.42.3.19	inx_free	344
5.42.3.20	inx_insert	344
5.42.3.21	inx_lock_read	344
5.42.3.22	inx_lock_write	345
5.42.3.23	inx_options	345
5.42.3.24	inx_replace	345
5.42.3.25	inx_serial	346
5.42.3.26	inx_truncate	346
5.42.3.27	inx_unlock	346
5.42.3.28	linked_alloc	346
5.42.4	Variable Documentation	347
5.42.4.1	inx_cursor_t	347
5.42.4.2	inx_t	347
5.43	src/core/indexes/inx.c File Reference	348
5.43.1	Function Documentation	349
5.43.1.1	inx_alloc	349
5.43.1.2	inx_append	349
5.43.1.3	inx_auto_read	349
5.43.1.4	inx_auto_unlock	350
5.43.1.5	inx_auto_write	350

5.43.1.6	<code>inx_cleanup</code>	350
5.43.1.7	<code>inx_count</code>	350
5.43.1.8	<code>inx_delete</code>	350
5.43.1.9	<code>inx_find</code>	351
5.43.1.10	<code>inx_free</code>	351
5.43.1.11	<code>inx_insert</code>	351
5.43.1.12	<code>inx_lock_read</code>	352
5.43.1.13	<code>inx_lock_write</code>	352
5.43.1.14	<code>inx_options</code>	353
5.43.1.15	<code>inx_replace</code>	353
5.43.1.16	<code>inx_serial</code>	353
5.43.1.17	<code>inx_truncate</code>	353
5.43.1.18	<code>inx_unlock</code>	353
5.44	<code>src/core/indexes/linked.c</code> File Reference	355
5.44.1	Function Documentation	356
5.44.1.1	<code>__attribute__</code>	356
5.44.1.2	<code>linked_alloc</code>	356
5.44.1.3	<code>linked_append</code>	357
5.44.1.4	<code>linked_cursor_active</code>	357
5.44.1.5	<code>linked_cursor_alloc</code>	357
5.44.1.6	<code>linked_cursor_free</code>	358
5.44.1.7	<code>linked_cursor_key_active</code>	358
5.44.1.8	<code>linked_cursor_key_next</code>	358
5.44.1.9	<code>linked_cursor_next</code>	359
5.44.1.10	<code>linked_cursor_reset</code>	359
5.44.1.11	<code>linked_cursor_value_active</code>	359
5.44.1.12	<code>linked_cursor_value_next</code>	359
5.44.1.13	<code>linked_delete</code>	360
5.44.1.14	<code>linked_find</code>	360
5.44.1.15	<code>linked_free</code>	360
5.44.1.16	<code>linked_insert</code>	361
5.44.1.17	<code>linked_record_alloc</code>	361
5.44.1.18	<code>linked_record_free</code>	362
5.44.1.19	<code>linked_record_get_data</code>	362
5.44.1.20	<code>linked_record_get_key</code>	362
5.44.1.21	<code>linked_truncate</code>	362
5.44.2	Variable Documentation	363
5.44.2.1	<code>linked_cursor_t</code>	363
5.44.2.2	<code>linked_node_t</code>	363

5.44.2.3	linked_record_t	363
5.45	src/core/memory/align.c File Reference	364
5.45.1	Function Documentation	364
5.45.1.1	align	364
5.46	src/core/memory/bitwise.c File Reference	365
5.46.1	Function Documentation	365
5.46.1.1	bitwise_and	365
5.46.1.2	bitwise_count	365
5.46.1.3	bitwise_or	365
5.46.1.4	bitwise_xor	366
5.47	src/core/parsers/bitwise.c File Reference	367
5.47.1	Function Documentation	367
5.47.1.1	st_and	367
5.47.1.2	st_bitwise	367
5.47.1.3	st_not	368
5.47.1.4	st_or	368
5.47.1.5	st_xor	369
5.48	src/core/memory/memory.c File Reference	370
5.48.1	Function Documentation	370
5.48.1.1	mm_alloc	370
5.48.1.2	mm_cleanup_variadic	371
5.48.1.3	mm_copy	371
5.48.1.4	mm_dupe	372
5.48.1.5	mm_empty	372
5.48.1.6	mm_free	372
5.48.1.7	mm_move	373
5.48.1.8	mm_set	373
5.48.1.9	mm_wipe	374
5.49	src/core/memory/memory.h File Reference	376
5.49.1	Define Documentation	377
5.49.1.1	MEMORYBUF	377
5.49.1.2	mm_cleanup	377
5.49.1.3	MM_SEC_PAGE_ALIGNMENT_MIN	378
5.49.1.4	MM_SEC_POOL_LENGTH_MIN	378
5.49.1.5	MM_SEC_REQUEST_ALIGNMENT	378
5.49.2	Function Documentation	378
5.49.2.1	align	378
5.49.2.2	bitwise_and	378
5.49.2.3	bitwise_count	378

5.49.2.4	<code>bitwise_or</code>	378
5.49.2.5	<code>bitwise_xor</code>	379
5.49.2.6	<code>mm_alloc</code>	379
5.49.2.7	<code>mm_cleanup_variadic</code>	379
5.49.2.8	<code>mm_copy</code>	380
5.49.2.9	<code>mm_dupe</code>	380
5.49.2.10	<code>mm_empty</code>	381
5.49.2.11	<code>mm_free</code>	381
5.49.2.12	<code>mm_move</code>	382
5.49.2.13	<code>mm_sec_alloc</code>	382
5.49.2.14	<code>mm_sec_cleanup</code>	382
5.49.2.15	<code>mm_sec_free</code>	383
5.49.2.16	<code>mm_sec_realloc</code>	383
5.49.2.17	<code>mm_sec_secured</code>	383
5.49.2.18	<code>mm_sec_start</code>	383
5.49.2.19	<code>mm_sec_stats</code>	384
5.49.2.20	<code>mm_sec_stop</code>	384
5.49.2.21	<code>mm_set</code>	384
5.49.2.22	<code>mm_wipe</code>	385
5.50	<code>src/core/memory/secure.c</code> File Reference	386
5.50.1	Enumeration Type Documentation	387
5.50.1.1	<code>"@7</code>	387
5.50.2	Function Documentation	387
5.50.2.1	<code>__attribute__</code>	387
5.50.2.2	<code>mm_sec_alloc</code>	387
5.50.2.3	<code>mm_sec_chunk_merge</code>	387
5.50.2.4	<code>mm_sec_chunk_new</code>	387
5.50.2.5	<code>mm_sec_chunk_next</code>	388
5.50.2.6	<code>mm_sec_chunk_prev</code>	388
5.50.2.7	<code>mm_sec_cleanup</code>	388
5.50.2.8	<code>mm_sec_free</code>	388
5.50.2.9	<code>mm_sec_realloc</code>	389
5.50.2.10	<code>mm_sec_secured</code>	389
5.50.2.11	<code>mm_sec_start</code>	389
5.50.2.12	<code>mm_sec_stats</code>	390
5.50.2.13	<code>mm_sec_stop</code>	390
5.50.3	Variable Documentation	390
5.50.3.1	<code>allocated</code>	390
5.50.3.2	<code>bytes</code>	390

5.50.3.3	data	390
5.50.3.4	data_true	391
5.50.3.5	enabled	391
5.50.3.6	items	391
5.50.3.7	length	391
5.50.3.8	length_true	391
5.50.3.9	lock	391
5.50.3.10	secured_t	391
5.50.3.11	slab	391
5.51	src/core/parsers/case.c File Reference	392
5.51.1	Function Documentation	392
5.51.1.1	lower_chr	392
5.51.1.2	lower_st	392
5.51.1.3	upper_chr	393
5.51.1.4	upper_st	393
5.52	src/core/parsers/formats/formats.h File Reference	394
5.52.1	Function Documentation	394
5.52.1.1	nvp_alloc	394
5.52.1.2	nvp_free	394
5.52.1.3	nvp_init	395
5.52.1.4	nvp_parse	395
5.53	src/core/parsers/formats/nvp.c File Reference	396
5.53.1	Function Documentation	396
5.53.1.1	nvp_alloc	396
5.53.1.2	nvp_free	396
5.53.1.3	nvp_parse	396
5.54	src/core/parsers/line.c File Reference	398
5.54.1	Function Documentation	398
5.54.1.1	line_pl_bl	398
5.54.1.2	line_pl_ns	398
5.54.1.3	line_pl_pl	399
5.54.1.4	line_pl_st	399
5.55	src/core/parsers/numbers/clamp.c File Reference	400
5.55.1	Function Documentation	400
5.55.1.1	int16_clamp	400
5.55.1.2	int32_clamp	401
5.55.1.3	int64_clamp	401
5.55.1.4	int8_clamp	401
5.55.1.5	uint16_clamp	402

5.55.1.6	uint32_clamp	402
5.55.1.7	uint64_clamp	402
5.55.1.8	uint8_clamp	403
5.56	src/core/parsers/numbers/digits.c File Reference	404
5.56.1	Function Documentation	404
5.56.1.1	int16_digits	404
5.56.1.2	int32_digits	404
5.56.1.3	int64_digits	405
5.56.1.4	int8_digits	405
5.56.1.5	uint16_digits	405
5.56.1.6	uint32_digits	405
5.56.1.7	uint64_digits	406
5.56.1.8	uint8_digits	406
5.57	src/core/parsers/numbers/numbers.c File Reference	407
5.57.1	Function Documentation	409
5.57.1.1	double_conv	409
5.57.1.2	float_conv	409
5.57.1.3	int16_conv_bl	409
5.57.1.4	int16_conv_ns	410
5.57.1.5	int16_conv_st	410
5.57.1.6	int32_conv_bl	410
5.57.1.7	int32_conv_ns	411
5.57.1.8	int32_conv_st	411
5.57.1.9	int64_conv_bl	412
5.57.1.10	int64_conv_ns	412
5.57.1.11	int64_conv_st	412
5.57.1.12	int8_conv_bl	413
5.57.1.13	int8_conv_ns	413
5.57.1.14	int8_conv_st	413
5.57.1.15	size_conv_bl	414
5.57.1.16	ssize_conv_bl	414
5.57.1.17	uint16_conv_bl	415
5.57.1.18	uint16_conv_ns	415
5.57.1.19	uint16_conv_st	415
5.57.1.20	uint16_get_no	416
5.57.1.21	uint16_put_no	416
5.57.1.22	uint24_get_no	416
5.57.1.23	uint24_put_no	416
5.57.1.24	uint32_conv_bl	417

5.57.1.25	uint32_conv_ns	417
5.57.1.26	uint32_conv_st	418
5.57.1.27	uint32_get_no	418
5.57.1.28	uint32_put_no	418
5.57.1.29	uint64_conv_bl	419
5.57.1.30	uint64_conv_ns	419
5.57.1.31	uint64_conv_pl	419
5.57.1.32	uint64_conv_st	420
5.57.1.33	uint8_conv_bl	420
5.57.1.34	uint8_conv_ns	421
5.57.1.35	uint8_conv_st	421
5.58	src/core/parsers/numbers/numbers.h File Reference	422
5.58.1	Function Documentation	425
5.58.1.1	double_conv	425
5.58.1.2	float_conv	425
5.58.1.3	int16_clamp	425
5.58.1.4	int16_conv_bl	426
5.58.1.5	int16_conv_ns	426
5.58.1.6	int16_conv_st	426
5.58.1.7	int16_digits	427
5.58.1.8	int32_clamp	427
5.58.1.9	int32_conv_bl	427
5.58.1.10	int32_conv_ns	428
5.58.1.11	int32_conv_st	428
5.58.1.12	int32_digits	428
5.58.1.13	int64_clamp	429
5.58.1.14	int64_conv_bl	429
5.58.1.15	int64_conv_ns	429
5.58.1.16	int64_conv_st	430
5.58.1.17	int64_digits	430
5.58.1.18	int8_clamp	430
5.58.1.19	int8_conv_bl	431
5.58.1.20	int8_conv_ns	431
5.58.1.21	int8_conv_st	431
5.58.1.22	int8_digits	432
5.58.1.23	size_conv_bl	432
5.58.1.24	ssize_conv_bl	432
5.58.1.25	uint16_clamp	433
5.58.1.26	uint16_conv_bl	433

5.58.1.27	uint16_conv_ns	433
5.58.1.28	uint16_conv_st	434
5.58.1.29	uint16_digits	434
5.58.1.30	uint16_get_no	434
5.58.1.31	uint16_put_no	434
5.58.1.32	uint24_get_no	435
5.58.1.33	uint24_put_no	435
5.58.1.34	uint32_clamp	435
5.58.1.35	uint32_conv_bl	436
5.58.1.36	uint32_conv_ns	436
5.58.1.37	uint32_conv_st	436
5.58.1.38	uint32_digits	437
5.58.1.39	uint32_get_no	437
5.58.1.40	uint32_put_no	437
5.58.1.41	uint64_clamp	437
5.58.1.42	uint64_conv_bl	438
5.58.1.43	uint64_conv_ns	438
5.58.1.44	uint64_conv_pl	439
5.58.1.45	uint64_conv_st	439
5.58.1.46	uint64_digits	439
5.58.1.47	uint8_clamp	440
5.58.1.48	uint8_conv_bl	440
5.58.1.49	uint8_conv_ns	440
5.58.1.50	uint8_conv_st	441
5.58.1.51	uint8_digits	441
5.59	src/core/parsers/parsers.h File Reference	442
5.59.1	Function Documentation	444
5.59.1.1	line_pl_bl	444
5.59.1.2	line_pl_ns	444
5.59.1.3	line_pl_pl	445
5.59.1.4	line_pl_st	445
5.59.1.5	lower_chr	445
5.59.1.6	lower_st	446
5.59.1.7	pl_get_embraced	446
5.59.1.8	pl_shrink_before_characters	446
5.59.1.9	pl_skip_characters	447
5.59.1.10	pl_skip_to_characters	447
5.59.1.11	pl_trim	447
5.59.1.12	pl_trim_end	447

5.59.1.13	pl_trim_start	448
5.59.1.14	pl_update_start	448
5.59.1.15	st_and	448
5.59.1.16	st_not	448
5.59.1.17	st_or	449
5.59.1.18	st_trim	449
5.59.1.19	st_xor	449
5.59.1.20	str_tok_get_bl	450
5.59.1.21	str_tok_get_count_bl	450
5.59.1.22	time_datestamp	450
5.59.1.23	time_print_gmt	451
5.59.1.24	time_print_local	451
5.59.1.25	time_till_midnight	451
5.59.1.26	tok_get_bl	452
5.59.1.27	tok_get_count_bl	452
5.59.1.28	tok_get_count_st	452
5.59.1.29	tok_get_ns	453
5.59.1.30	tok_get_pl	453
5.59.1.31	tok_get_st	454
5.59.1.32	tok_pop	454
5.59.1.33	tok_pop_init_bl	454
5.59.1.34	tok_pop_init_st	454
5.59.1.35	upper_chr	454
5.59.1.36	upper_st	455
5.60	src/providers/parsers/parsers.h File Reference	456
5.60.1	Function Documentation	458
5.60.1.1	lib_load_jansson	458
5.60.1.2	lib_load_utf8proc	458
5.60.1.3	lib_load_xml	459
5.60.1.4	lib_version_jansson	459
5.60.1.5	lib_version_utf8proc	459
5.60.1.6	lib_version_xml	459
5.60.1.7	utf8_error_string	459
5.60.1.8	utf8_length_st	460
5.60.1.9	utf8_valid_st	460
5.60.1.10	xml_create_doc	460
5.60.1.11	xml_create_parser_ctx	461
5.60.1.12	xml_create_xpath_ctx	461
5.60.1.13	xml_dump_doc	461

5.60.1.14	xml_encode	462
5.60.1.15	xml_error	462
5.60.1.16	xml_free_doc	462
5.60.1.17	xml_free_parser_ctx	463
5.60.1.18	xml_free_xpath_ctx	463
5.60.1.19	xml_free_xpath_obj	463
5.60.1.20	xml_get_xpath_int16	463
5.60.1.21	xml_get_xpath_int32	464
5.60.1.22	xml_get_xpath_int64	464
5.60.1.23	xml_get_xpath_int8	464
5.60.1.24	xml_get_xpath_node_count	465
5.60.1.25	xml_get_xpath_ns	465
5.60.1.26	xml_get_xpath_st	465
5.60.1.27	xml_get_xpath_uint16	465
5.60.1.28	xml_get_xpath_uint32	466
5.60.1.29	xml_get_xpath_uint64	466
5.60.1.30	xml_get_xpath_uint8	466
5.60.1.31	xml_node_add_sibling	466
5.60.1.32	xml_node_free	467
5.60.1.33	xml_node_get_content_st	467
5.60.1.34	xml_node_new	467
5.60.1.35	xml_node_set_content	468
5.60.1.36	xml_node_set_property	468
5.60.1.37	xml_set_xpath_ns	468
5.60.1.38	xml_set_xpath_property	469
5.60.1.39	xml_set_xpath_uint64	469
5.60.1.40	xml_start	469
5.60.1.41	xml_stop	469
5.60.1.42	xml_xpath_eval	470
5.60.1.43	xml_xpath_set_namespace	470
5.61	src/core/parsers/special/bracket.c File Reference	471
5.61.1	Function Documentation	471
5.61.1.1	bracket_extract_pl	471
5.62	src/core/parsers/special/special.h File Reference	472
5.62.1	Function Documentation	472
5.62.1.1	bracket_extract_pl	472
5.63	src/core/parsers/time.c File Reference	473
5.63.1	Function Documentation	473
5.63.1.1	time_datestamp	473

5.63.1.2	time_print_gmt	473
5.63.1.3	time_print_local	474
5.63.1.4	time_till_midnight	474
5.64	src/core/parsers/token.c File Reference	475
5.64.1	Function Documentation	476
5.64.1.1	pl_get_embraced	476
5.64.1.2	pl_shrink_before_characters	476
5.64.1.3	pl_skip_characters	476
5.64.1.4	pl_skip_to_characters	477
5.64.1.5	pl_update_start	477
5.64.1.6	str_tok_get_bl	477
5.64.1.7	str_tok_get_count_bl	478
5.64.1.8	tok_get_bl	478
5.64.1.9	tok_get_count_bl	478
5.64.1.10	tok_get_count_st	479
5.64.1.11	tok_get_ns	479
5.64.1.12	tok_get_pl	480
5.64.1.13	tok_get_st	480
5.64.1.14	tok_pop	480
5.64.1.15	tok_pop_init_bl	481
5.64.1.16	tok_pop_init_st	481
5.65	src/core/parsers/trim.c File Reference	482
5.65.1	Function Documentation	482
5.65.1.1	pl_trim	482
5.65.1.2	pl_trim_end	482
5.65.1.3	pl_trim_start	482
5.65.1.4	st_trim	482
5.66	src/core/strings/allocation.c File Reference	484
5.66.1	Function Documentation	485
5.66.1.1	st_alloc	485
5.66.1.2	st_alloc_opts	485
5.66.1.3	st_append_opts	486
5.66.1.4	st_append_out	486
5.66.1.5	st_cleanup_variadic	486
5.66.1.6	st_copy_in	487
5.66.1.7	st_dupe	487
5.66.1.8	st_dupe_opts	487
5.66.1.9	st_free	488
5.66.1.10	st_import	489

5.66.1.11	st_import_opts	490
5.66.1.12	st_merge_opts	490
5.66.1.13	st_nullify	491
5.66.1.14	st_output	491
5.66.1.15	st_realloc	491
5.67	src/core/strings/data.c File Reference	493
5.67.1	Function Documentation	493
5.67.1.1	st_char_get	493
5.67.1.2	st_data_get	495
5.67.1.3	st_data_set	495
5.67.1.4	st_empty_out	496
5.67.1.5	st_empty_variadic	496
5.67.1.6	st_populated_variadic	497
5.67.1.7	st_set	497
5.67.1.8	st_uchar_get	497
5.67.1.9	st_wipe	498
5.68	src/providers/storage/data.c File Reference	499
5.68.1	Function Documentation	499
5.68.1.1	tank_delete_object	499
5.68.1.2	tank_insert_object	499
5.69	src/servers/http/data.c File Reference	500
5.69.1	Function Documentation	500
5.69.1.1	http_data_free	500
5.69.1.2	http_data_get	500
5.69.1.3	http_data_header_parse	501
5.69.1.4	http_data_header_parse_line	501
5.69.1.5	http_data_value_decode	501
5.69.1.6	http_data_value_parse	502
5.70	src/core/strings/info.c File Reference	503
5.70.1	Function Documentation	503
5.70.1.1	st_info_allocator	503
5.70.1.2	st_info_layout	503
5.70.1.3	st_info_opts	504
5.70.1.4	st_info_type	504
5.70.2	Variable Documentation	504
5.70.2.1	st_option_allocators	504
5.70.2.2	st_option_flags	505
5.70.2.3	st_option_layouts	505
5.70.2.4	st_option_types	505

5.71	src/core/strings/length.c File Reference	506
5.71.1	Function Documentation	506
5.71.1.1	st_avail_get	506
5.71.1.2	st_avail_set	506
5.71.1.3	st_length_get	507
5.71.1.4	st_length_int	508
5.71.1.5	st_length_set	509
5.72	src/core/strings/multi.c File Reference	510
5.72.1	Function Documentation	510
5.72.1.1	cmp_mt_mt	510
5.72.1.2	ident_mt_mt	511
5.72.1.3	mt_dupe	511
5.72.1.4	mt_free	512
5.72.1.5	mt_get_char	512
5.72.1.6	mt_get_length	512
5.72.1.7	mt_get_null	513
5.72.1.8	mt_get_number	513
5.72.1.9	mt_get_type	513
5.72.1.10	mt_is_empty	514
5.72.1.11	mt_is_number	514
5.72.1.12	mt_set_type	514
5.73	src/core/strings/nuller.c File Reference	516
5.73.1	Function Documentation	516
5.73.1.1	ns_alloc	516
5.73.1.2	ns_append	517
5.73.1.3	ns_cleanup_variadic	517
5.73.1.4	ns_dupe	517
5.73.1.5	ns_empty	518
5.73.1.6	ns_empty_out	518
5.73.1.7	ns_free	518
5.73.1.8	ns_import	519
5.73.1.9	ns_length_get	519
5.73.1.10	ns_length_int	519
5.73.1.11	ns_populated_variadic	520
5.73.1.12	ns_wipe	520
5.74	src/core/strings/opts.c File Reference	521
5.74.1	Function Documentation	521
5.74.1.1	st_opt_get	521
5.74.1.2	st_opt_set	521

5.74.1.3	st_opt_test	522
5.75	src/core/strings/print.c File Reference	523
5.75.1	Function Documentation	523
5.75.1.1	st_aprint	523
5.75.1.2	st_aprint_opts	524
5.75.1.3	st_quick	524
5.75.1.4	st_sprint	524
5.75.1.5	st_vaprint_opts	525
5.75.1.6	st_vsprint	525
5.75.1.7	st_write_variadic	526
5.76	src/core/strings/replace.c File Reference	527
5.76.1	Function Documentation	527
5.76.1.1	st_replace	527
5.76.1.2	st_swap	527
5.77	src/core/strings/shortcuts.c File Reference	528
5.77.1	Function Documentation	528
5.77.1.1	pl_char_get	528
5.77.1.2	pl_clone	529
5.77.1.3	pl_data_get	529
5.77.1.4	pl_empty	529
5.77.1.5	pl_inc	530
5.77.1.6	pl_init	530
5.77.1.7	pl_length_get	530
5.77.1.8	pl_length_int	531
5.77.1.9	pl_null	531
5.77.1.10	pl_set	531
5.77.1.11	pl_starts_with_char	532
5.78	src/core/strings/strings.h File Reference	533
5.78.1	Define Documentation	539
5.78.1.1	__VA_NARG__	539
5.78.1.2	__VA_NARG_N	539
5.78.1.3	__VA_NARG_SEQ_N	539
5.78.1.4	BLOCK	539
5.78.1.5	BLOCKBUF	539
5.78.1.6	CONSTANT	539
5.78.1.7	MANAGED	539
5.78.1.8	MANAGEDBUF	539
5.78.1.9	ns_cleanup	540
5.78.1.10	ns_populated	540

5.78.1.11 NULLER	540
5.78.1.12 PLACER	540
5.78.1.13 st_append	541
5.78.1.14 st_cleanup	541
5.78.1.15 st_empty	542
5.78.1.16 st_merge	542
5.78.1.17 st_populated	542
5.78.1.18 st_vaprint	542
5.78.1.19 st_write	543
5.78.1.20 va_narg	543
5.78.2 Typedef Documentation	543
5.78.2.1 stringer_t	543
5.78.3 Enumeration Type Documentation	543
5.78.3.1 "@11	543
5.78.4 Function Documentation	543
5.78.4.1 __attribute__	543
5.78.4.2 cmp_mt_mt	544
5.78.4.3 ident_mt_mt	544
5.78.4.4 mt_dupe	544
5.78.4.5 mt_free	545
5.78.4.6 mt_get_char	545
5.78.4.7 mt_get_length	545
5.78.4.8 mt_get_null	546
5.78.4.9 mt_get_number	546
5.78.4.10 mt_get_type	546
5.78.4.11 mt_is_empty	547
5.78.4.12 mt_is_number	547
5.78.4.13 mt_set_type	547
5.78.4.14 ns_alloc	548
5.78.4.15 ns_append	548
5.78.4.16 ns_cleanup_variadic	548
5.78.4.17 ns_dupe	549
5.78.4.18 ns_empty	549
5.78.4.19 ns_empty_out	549
5.78.4.20 ns_free	550
5.78.4.21 ns_import	550
5.78.4.22 ns_length_get	550
5.78.4.23 ns_length_int	551
5.78.4.24 ns_populated_variadic	551

5.78.4.25 ns_wipe	551
5.78.4.26 pl_char_get	552
5.78.4.27 pl_clone	552
5.78.4.28 pl_data_get	552
5.78.4.29 pl_empty	553
5.78.4.30 pl_inc	553
5.78.4.31 pl_init	553
5.78.4.32 pl_length_get	554
5.78.4.33 pl_length_int	554
5.78.4.34 pl_null	554
5.78.4.35 pl_set	555
5.78.4.36 pl_starts_with_char	555
5.78.4.37 st_alloc	555
5.78.4.38 st_alloc_opts	556
5.78.4.39 st_append_opts	556
5.78.4.40 st_append_out	557
5.78.4.41 st_aprint	557
5.78.4.42 st_aprint_opts	557
5.78.4.43 st_avail_get	557
5.78.4.44 st_avail_set	558
5.78.4.45 st_char_get	558
5.78.4.46 st_cleanup_variadic	559
5.78.4.47 st_copy_in	559
5.78.4.48 st_data_get	560
5.78.4.49 st_data_set	561
5.78.4.50 st_dupe	561
5.78.4.51 st_dupe_opts	561
5.78.4.52 st_empty_out	562
5.78.4.53 st_empty_variadic	563
5.78.4.54 st_free	563
5.78.4.55 st_import	564
5.78.4.56 st_import_opts	565
5.78.4.57 st_info_allocator	565
5.78.4.58 st_info_layout	565
5.78.4.59 st_info_opts	566
5.78.4.60 st_info_type	566
5.78.4.61 st_length_get	566
5.78.4.62 st_length_int	568
5.78.4.63 st_length_set	568

5.78.4.64	st_merge_opts	569
5.78.4.65	st_nullify	569
5.78.4.66	st_opt_get	570
5.78.4.67	st_opt_set	570
5.78.4.68	st_opt_test	570
5.78.4.69	st_output	571
5.78.4.70	st_populated_variadic	571
5.78.4.71	st_quick	571
5.78.4.72	st_realloc	571
5.78.4.73	st_replace	572
5.78.4.74	st_set	572
5.78.4.75	st_sprint	572
5.78.4.76	st_swap	572
5.78.4.77	st_uchar_get	573
5.78.4.78	st_used_variadic	573
5.78.4.79	st_valid_append	573
5.78.4.80	st_valid_avail	573
5.78.4.81	st_valid_destination	574
5.78.4.82	st_valid_free	574
5.78.4.83	st_valid_jointed	574
5.78.4.84	st_valid_opts	575
5.78.4.85	st_valid_placer	575
5.78.4.86	st_valid_tracked	576
5.78.4.87	st_vaprint_opts	576
5.78.4.88	st_vsprint	576
5.78.4.89	st_wipe	577
5.78.4.90	st_write_variadic	577
5.78.5	Variable Documentation	577
5.78.5.1	block_t	577
5.78.5.2	constant_t	578
5.78.5.3	managed_t	578
5.78.5.4	mapped_t	578
5.78.5.5	nuller_t	578
5.78.5.6	placer_t	578
5.79	src/core/strings/validate.c File Reference	579
5.79.1	Function Documentation	579
5.79.1.1	st_valid_append	579
5.79.1.2	st_valid_avail	580
5.79.1.3	st_valid_destination	580

5.79.1.4	st_valid_free	580
5.79.1.5	st_valid_joined	581
5.79.1.6	st_valid_opts	581
5.79.1.7	st_valid_placer	581
5.79.1.8	st_valid_tracked	582
5.80	src/core/thread/keys.c File Reference	583
5.80.1	Function Documentation	583
5.80.1.1	tkey_get	583
5.80.1.2	tkey_init	583
5.80.1.3	tkey_set	584
5.81	src/providers/dime/signet/keys.c File Reference	585
5.81.1	Function Documentation	585
5.81.1.1	dime_keys_enckey_fetch	585
5.81.1.2	dime_keys_file_create	585
5.81.1.3	dime_keys_generate	586
5.81.1.4	dime_keys_signkey_fetch	586
5.82	src/core/thread/mutex.c File Reference	587
5.82.1	Function Documentation	587
5.82.1.1	mutex_destroy	587
5.82.1.2	mutex_init	587
5.82.1.3	mutex_lock	588
5.82.1.4	mutex_unlock	588
5.83	src/core/thread/rwlock.c File Reference	590
5.83.1	Function Documentation	590
5.83.1.1	rwlock_attr_destroy	590
5.83.1.2	rwlock_attr_getkind	591
5.83.1.3	rwlock_attr_init	591
5.83.1.4	rwlock_attr_setkind	591
5.83.1.5	rwlock_destroy	592
5.83.1.6	rwlock_init	592
5.83.1.7	rwlock_lock_read	592
5.83.1.8	rwlock_lock_write	593
5.83.1.9	rwlock_unlock	593
5.84	src/core/thread/thread.c File Reference	594
5.84.1	Function Documentation	594
5.84.1.1	thread_alloc	594
5.84.1.2	thread_cancel	595
5.84.1.3	thread_cancel_disable	595
5.84.1.4	thread_cancel_enable	595

5.84.1.5	thread_get_thread_id	595
5.84.1.6	thread_join	595
5.84.1.7	thread_launch	596
5.84.1.8	thread_result	596
5.84.1.9	thread_signal	597
5.85	src/engine/context/thread.c File Reference	598
5.85.1	Function Documentation	598
5.85.1.1	thread_start	598
5.85.1.2	thread_stop	598
5.86	src/core/thread/thread.h File Reference	599
5.86.1	Function Documentation	600
5.86.1.1	mutex_destroy	600
5.86.1.2	mutex_init	601
5.86.1.3	mutex_lock	601
5.86.1.4	mutex_unlock	601
5.86.1.5	rwlock_attr_destroy	602
5.86.1.6	rwlock_attr_getkind	602
5.86.1.7	rwlock_attr_init	603
5.86.1.8	rwlock_attr_setkind	603
5.86.1.9	rwlock_destroy	603
5.86.1.10	rwlock_init	604
5.86.1.11	rwlock_lock_read	604
5.86.1.12	rwlock_lock_write	605
5.86.1.13	rwlock_unlock	605
5.86.1.14	thread_alloc	605
5.86.1.15	thread_cancel	606
5.86.1.16	thread_cancel_disable	606
5.86.1.17	thread_cancel_enable	606
5.86.1.18	thread_get_thread_id	606
5.86.1.19	thread_join	607
5.86.1.20	thread_launch	607
5.86.1.21	thread_result	607
5.86.1.22	thread_signal	608
5.86.1.23	tkey_get	608
5.86.1.24	tkey_init	608
5.86.1.25	tkey_set	609
5.87	src/core/type.c File Reference	610
5.87.1	Function Documentation	610
5.87.1.1	type	610

5.88	src/engine/config/cache/cache.c File Reference	611
5.88.1	Function Documentation	611
5.88.1.1	cache_alloc	611
5.88.1.2	cache_config	611
5.88.1.3	cache_free	612
5.88.1.4	cache_output_help	612
5.88.1.5	cache_output_settings	613
5.88.1.6	cache_set_value	613
5.88.1.7	cache_validate	613
5.89	src/objects/mail/cache.c File Reference	615
5.89.1	Function Documentation	615
5.89.1.1	mail_cache_destroy	615
5.89.1.2	mail_cache_get	615
5.89.1.3	mail_cache_reset	616
5.89.1.4	mail_cache_set	616
5.89.1.5	mail_cache_start	616
5.89.1.6	mail_cache_stop	617
5.89.1.7	mail_cache_thread_stop	617
5.90	src/providers/consumers/cache.c File Reference	618
5.90.1	Function Documentation	619
5.90.1.1	cache_add	619
5.90.1.2	cache_append	619
5.90.1.3	cache_decrement	619
5.90.1.4	cache_delete	620
5.90.1.5	cache_flush	620
5.90.1.6	cache_get	620
5.90.1.7	cache_get_u64	620
5.90.1.8	cache_increment	621
5.90.1.9	cache_set	621
5.90.1.10	cache_set_u64	622
5.90.1.11	cache_silent_add	622
5.90.1.12	cache_start	622
5.90.1.13	cache_stop	622
5.90.1.14	lib_load_cache	623
5.90.1.15	lib_version_cache	623
5.90.2	Variable Documentation	623
5.90.2.1	cache_pool	623
5.91	src/providers/dime/signet-resolver/cache.c File Reference	624
5.91.1	Function Documentation	627

5.91.1.1	_add_cached_object	627
5.91.1.2	_add_cached_object_cmp	627
5.91.1.3	_add_cached_object_cmp_forced	628
5.91.1.4	_add_cached_object_forced	628
5.91.1.5	_cached_object_exists	629
5.91.1.6	_cached_object_exists_cmp	629
5.91.1.7	_clone_cached_object	630
5.91.1.8	_create_cached_object	630
5.91.1.9	_deserialize_array	630
5.91.1.10	_deserialize_array_cb	631
5.91.1.11	_deserialize_data	631
5.91.1.12	_deserialize_signet_cb	632
5.91.1.13	_deserialize_str_array	632
5.91.1.14	_deserialize_string	633
5.91.1.15	_deserialize_vardata	633
5.91.1.16	_destroy_cache_entry	633
5.91.1.17	_destroy_signet_cb	634
5.91.1.18	_dump_cache	634
5.91.1.19	_dump_cache_data	634
5.91.1.20	_dump_signet_cb	634
5.91.1.21	_evict_if_stale	635
5.91.1.22	_find_cached_object	635
5.91.1.23	_find_cached_object_cmp	635
5.91.1.24	_get_cache_location	636
5.91.1.25	_get_cache_obj_data	636
5.91.1.26	_get_cached_store_by_type	637
5.91.1.27	_get_dime_dir_location	637
5.91.1.28	_is_object_expired	637
5.91.1.29	_load_cache_contents	638
5.91.1.30	_lock_cache_store	638
5.91.1.31	_mem_append_serialized	638
5.91.1.32	_mem_append_serialized_array	638
5.91.1.33	_mem_append_serialized_array_cb	639
5.91.1.34	_mem_append_serialized_str_array	639
5.91.1.35	_mem_append_serialized_string	640
5.91.1.36	_remove_cached_object	640
5.91.1.37	_remove_cached_object_cmp	640
5.91.1.38	_replace_object	641
5.91.1.39	_save_cache_contents	641

5.91.1.40	<code>_serialize_signet_cb</code>	641
5.91.1.41	<code>_set_cache_location</code>	642
5.91.1.42	<code>_set_cache_permissions</code>	642
5.91.1.43	<code>_unlink_object</code>	642
5.91.1.44	<code>_unlock_cache_store</code>	643
5.91.2	Variable Documentation	643
5.91.2.1	<code>cached_stores</code>	643
5.92	<code>src/engine/config/cache/cache.h</code> File Reference	644
5.92.1	Function Documentation	644
5.92.1.1	<code>cache_alloc</code>	644
5.92.1.2	<code>cache_config</code>	645
5.92.1.3	<code>cache_free</code>	645
5.92.1.4	<code>cache_output_help</code>	645
5.92.1.5	<code>cache_output_settings</code>	646
5.92.1.6	<code>cache_set_value</code>	646
5.92.1.7	<code>cache_validate</code>	646
5.93	<code>src/providers/dime/signet-resolver/cache.h</code> File Reference	648
5.93.1	Define Documentation	651
5.93.1.1	<code>CACHE_PERM_ADD</code>	651
5.93.1.2	<code>CACHE_PERM_ALL_FLAGS</code>	651
5.93.1.3	<code>CACHE_PERM_DEFAULT</code>	651
5.93.1.4	<code>CACHE_PERM_DELETE</code>	651
5.93.1.5	<code>CACHE_PERM_LOAD</code>	651
5.93.1.6	<code>CACHE_PERM_NONE</code>	651
5.93.1.7	<code>CACHE_PERM_READ</code>	651
5.93.1.8	<code>CACHE_PERM_SAVE</code>	651
5.93.2	Typedef Documentation	651
5.93.2.1	<code>cached_object_t</code>	651
5.93.2.2	<code>cached_store_comparator_t</code>	651
5.93.2.3	<code>custom_deserializer_t</code>	652
5.93.2.4	<code>custom_dumper_t</code>	652
5.93.2.5	<code>custom_serializer_t</code>	652
5.93.3	Enumeration Type Documentation	652
5.93.3.1	<code>cached_data_type_t</code>	652
5.93.4	Function Documentation	652
5.93.4.1	<code>_clone_cached_object</code>	652
5.93.4.2	<code>_create_cached_object</code>	652
5.93.4.3	<code>_deserialize_array</code>	653
5.93.4.4	<code>_deserialize_array_cb</code>	653

5.93.4.5	_deserialize_data	654
5.93.4.6	_deserialize_signet_cb	654
5.93.4.7	_deserialize_str_array	655
5.93.4.8	_deserialize_string	655
5.93.4.9	_deserialize_vardata	655
5.93.4.10	_destroy_signet_cb	656
5.93.4.11	_dump_cache	656
5.93.4.12	_dump_cache_data	656
5.93.4.13	_dump_signet_cb	657
5.93.4.14	_evict_if_stale	657
5.93.4.15	_get_cached_store_by_type	657
5.93.4.16	_is_object_expired	657
5.93.4.17	_lock_cache_store	658
5.93.4.18	_mem_append_serialized	658
5.93.4.19	_mem_append_serialized_array	659
5.93.4.20	_mem_append_serialized_array_cb	659
5.93.4.21	_mem_append_serialized_str_array	659
5.93.4.22	_mem_append_serialized_string	660
5.93.4.23	_replace_object	660
5.93.4.24	_serialize_signet_cb	660
5.93.4.25	_unlink_object	661
5.93.4.26	_unlock_cache_store	661
5.93.4.27	PUBLIC_FUNC_DECL	662
5.93.4.28	PUBLIC_FUNC_DECL	662
5.93.4.29	PUBLIC_FUNC_DECL	662
5.93.4.30	PUBLIC_FUNC_DECL	662
5.93.4.31	PUBLIC_FUNC_DECL	662
5.93.4.32	PUBLIC_FUNC_DECL	662
5.93.4.33	PUBLIC_FUNC_DECL	662
5.93.4.34	PUBLIC_FUNC_DECL	662
5.93.4.35	PUBLIC_FUNC_DECL	662
5.93.4.36	PUBLIC_FUNC_DECL	662
5.93.4.37	PUBLIC_FUNC_DECL	662
5.93.4.38	PUBLIC_FUNC_DECL	662
5.93.4.39	PUBLIC_FUNC_DECL	662
5.93.4.40	PUBLIC_FUNC_DECL	662
5.93.4.41	PUBLIC_FUNC_DECL	662
5.93.4.42	PUBLIC_FUNC_DECL	662
5.93.4.43	PUBLIC_FUNC_DECL	662

5.93.4.44	PUBLIC_FUNC_DECL	662
5.93.5	Variable Documentation	662
5.93.5.1	cached_stores	662
5.94	src/engine/config/cache/keys.h File Reference	663
5.94.1	Variable Documentation	663
5.94.1.1	cache_keys	663
5.95	src/engine/config/global/keys.h File Reference	664
5.95.1	Variable Documentation	664
5.95.1.1	magma_keys	664
5.96	src/engine/config/relay/keys.h File Reference	665
5.96.1	Variable Documentation	665
5.96.1.1	relay_keys	665
5.97	src/engine/config/servers/keys.h File Reference	666
5.97.1	Variable Documentation	666
5.97.1.1	server_keys	666
5.98	src/providers/dime/signet/keys.h File Reference	667
5.98.1	Enumeration Type Documentation	667
5.98.1.1	keys_type_t	667
5.98.2	Function Documentation	667
5.98.2.1	dime_keys_enckey_fetch	667
5.98.2.2	dime_keys_file_create	668
5.98.2.3	dime_keys_generate	668
5.98.2.4	dime_keys_signkey_fetch	668
5.99	src/providers/prime/keys/keys.h File Reference	669
5.99.1	Function Documentation	669
5.99.1.1	org_encrypted_key_get	669
5.99.1.2	org_encrypted_key_set	669
5.99.1.3	org_key_alloc	669
5.99.1.4	org_key_free	670
5.99.1.5	org_key_generate	670
5.99.1.6	org_key_get	670
5.99.1.7	org_key_length	670
5.99.1.8	org_key_set	670
5.99.1.9	user_encrypted_key_get	670
5.99.1.10	user_encrypted_key_set	670
5.99.1.11	user_key_alloc	671
5.99.1.12	user_key_free	671
5.99.1.13	user_key_generate	671
5.99.1.14	user_key_get	671

5.99.1.15	user_key_length	671
5.99.1.16	user_key_set	671
5.100	src/engine/config/config.h File Reference	672
5.100.1	Define Documentation	672
5.100.1.1	CONFIG_CHECK_DIR_READABLE	672
5.100.1.2	CONFIG_CHECK_DIR_READWRITE	673
5.100.1.3	CONFIG_CHECK_EXISTS	673
5.100.1.4	CONFIG_CHECK_FILE_READABLE	673
5.100.1.5	CONFIG_CHECK_FILE_READWRITE	673
5.100.1.6	CONFIG_CHECK_READWRITE	673
5.100.1.7	MAGMA_BLACKLIST_INSTANCES	673
5.100.1.8	MAGMA_CACHE_INSTANCES	673
5.100.1.9	MAGMA_CACHE_SERVER_RETRY	674
5.100.1.10	MAGMA_CACHE_SOCKET_TIMEOUT	674
5.100.1.11	MAGMA_CONNECTION_BUFFER_SIZE	674
5.100.1.12	MAGMA_CRYPTOGRAPHY_SEED_SIZE	674
5.100.1.13	MAGMA_FILENAME_MAX	674
5.100.1.14	MAGMA_FILEPATH_MAX	674
5.100.1.15	MAGMA_HOSTNAME_MAX	674
5.100.1.16	MAGMA_LOCATION_CACHE	674
5.100.1.17	MAGMA_LOGS	674
5.100.1.18	MAGMA_RELAY_INSTANCES	674
5.100.1.19	MAGMA_RESOURCE_FONTS	674
5.100.1.20	MAGMA_RESOURCE_LOCATION	675
5.100.1.21	MAGMA_RESOURCE_PAGES	675
5.100.1.22	MAGMA_RESOURCE_TEMPLATES	675
5.100.1.23	MAGMA_RESOURCE_VIRUS	675
5.100.1.24	MAGMA_SERVER_INSTANCES	675
5.100.1.25	MAGMA_SMTP_LINE_WRAP_LENGTH	675
5.100.1.26	MAGMA_SMTP_MAX_ADDRESS_SIZE	675
5.100.1.27	MAGMA_SMTP_MAX_HELO_SIZE	675
5.100.1.28	MAGMA_SMTP_MAX_MESSAGE_SIZE	675
5.100.1.29	MAGMA_SMTP_RECIPIENT_LIMIT	675
5.100.1.30	MAGMA_SMTP_RELAY_LIMIT	675
5.100.1.31	MAGMA_THREAD_BUFFER_SIZE	676
5.100.1.32	MAGMA_THREAD_STACK_SIZE	676
5.100.1.33	MAGMA_WORKER_THREAD_LIMIT	676
5.101	src/objects/config/config.h File Reference	677
5.101.1	Enumeration Type Documentation	677

5.101.1.1 "@66	677
5.101.2 Function Documentation	678
5.101.2.1 __attribute__	678
5.101.2.2 user_config_alloc	678
5.101.2.3 user_config_create	678
5.101.2.4 user_config_delete	678
5.101.2.5 user_config_edit	679
5.101.2.6 user_config_entry_alloc	679
5.101.2.7 user_config_entry_free	679
5.101.2.8 user_config_fetch	680
5.101.2.9 user_config_free	680
5.101.2.10 user_config_update	680
5.101.2.11 user_config_upsert	681
5.101.3 Variable Documentation	681
5.101.3.1 user_config_entry_t	681
5.101.3.2 user_config_t	681
5.102src/engine/config/global/datatier.c File Reference	682
5.102.1 Function Documentation	682
5.102.1.1 config_fetch_host_number	682
5.102.1.2 config_fetch_settings	682
5.103src/objects/auth/datatier.c File Reference	683
5.103.1 Function Documentation	683
5.103.1.1 auth_data_fetch	683
5.103.1.2 auth_data_update_legacy	683
5.103.1.3 auth_data_update_lock	684
5.104src/objects/config/datatier.c File Reference	685
5.104.1 Function Documentation	685
5.104.1.1 user_config_delete	685
5.104.1.2 user_config_fetch	685
5.104.1.3 user_config_upsert	686
5.105src/objects/contacts/datatier.c File Reference	687
5.105.1 Function Documentation	687
5.105.1.1 contact_delete	687
5.105.1.2 contact_detail_delete	688
5.105.1.3 contact_detail_upsert	688
5.105.1.4 contact_details_fetch	688
5.105.1.5 contact_insert	689
5.105.1.6 contact_update	689
5.105.1.7 contact_update_stamp	689

5.105.1.8 contacts_fetch	690
5.106src/objects/folders/datatier.c File Reference	691
5.106.1 Function Documentation	691
5.106.1.1 magma_folder_delete	691
5.106.1.2 magma_folder_fetch	691
5.106.1.3 magma_folder_insert	692
5.106.1.4 magma_folder_rename	692
5.107src/objects/mail/datatier.c File Reference	693
5.107.1 Function Documentation	693
5.107.1.1 mail_db_delete_message	693
5.107.1.2 mail_db_hide_message	693
5.107.1.3 mail_db_insert_duplicate_message	694
5.107.1.4 mail_db_insert_message	694
5.107.1.5 mail_db_update_message_folder	695
5.108src/objects/messages/datatier.c File Reference	696
5.108.1 Function Documentation	696
5.108.1.1 meta_data_fetch_folder_messages	696
5.108.1.2 meta_data_fetch_message_tags	696
5.108.1.3 meta_data_fetch_messages	697
5.109src/objects/meta/datatier.c File Reference	698
5.109.1 Function Documentation	699
5.109.1.1 meta_data_acknowledge_alert	699
5.109.1.2 meta_data_delete_folder	699
5.109.1.3 meta_data_delete_tag	700
5.109.1.4 meta_data_fetch_alerts	700
5.109.1.5 meta_data_fetch_all_tags	700
5.109.1.6 meta_data_fetch_folders	701
5.109.1.7 meta_data_fetch_keys	701
5.109.1.8 meta_data_fetch_mailbox_aliases	701
5.109.1.9 meta_data_fetch_shard	702
5.109.1.10meta_data_fetch_user	702
5.109.1.11meta_data_flags_add	702
5.109.1.12meta_data_flags_remove	703
5.109.1.13meta_data_flags_replace	703
5.109.1.14meta_data_insert_folder	704
5.109.1.15meta_data_insert_keys	704
5.109.1.16meta_data_insert_shard	704
5.109.1.17meta_data_insert_tag	705
5.109.1.18meta_data_truncate_tags	705

5.109.1.19	meta_data_update_folder_name	705
5.109.1.20	meta_data_update_log	706
5.110	src/objects/warehouse/datatier.c File Reference	707
5.110.1	Function Documentation	707
5.110.1.1	warehouse_fetch_domains	707
5.110.1.2	warehouse_fetch_patterns	707
5.111	src/servers/smtp/datatier.c File Reference	708
5.111.1	Function Documentation	708
5.111.1.1	smtp_check_authorized_from	708
5.111.1.2	smtp_check_receive_quota	709
5.111.1.3	smtp_check_transmit_quota	709
5.111.1.4	smtp_fetch_authorization	710
5.111.1.5	smtp_fetch_autoreply	710
5.111.1.6	smtp_fetch_inbound	710
5.111.1.7	smtp_fetch_rollmessages	712
5.111.1.8	smtp_get_action	712
5.111.1.9	smtp_insert_spamsig	712
5.111.1.10	smtp_update_receive_stats	713
5.111.1.11	smtp_update_transmission_stats	713
5.112	src/web/register/datatier.c File Reference	715
5.112.1	Function Documentation	715
5.112.1.1	register_data_check_username	715
5.112.1.2	register_data_fetch_blocklist	715
5.112.1.3	register_data_insert_user	716
5.113	src/web/statistics/datatier.c File Reference	717
5.113.1	Define Documentation	717
5.113.1.1	PORTAL_STATISTICS_TIMEOUT	717
5.113.2	Function Documentation	717
5.113.2.1	statistics_init	717
5.113.2.2	statistics_refresh	718
5.113.3	Variable Documentation	718
5.113.3.1	portal_statistics_mutex	718
5.113.3.2	portal_stats	718
5.113.3.3	statistics_last_updated	718
5.114	src/web/teacher/datatier.c File Reference	719
5.114.1	Function Documentation	719
5.114.1.1	teacher_data_delete	719
5.114.1.2	teacher_data_fetch	720
5.114.1.3	teacher_data_free	720

5.114.1.4	teacher_data_get	720
5.114.1.5	teacher_data_save	720
5.115	src/engine/config/global/global.c File Reference	722
5.115.1	Function Documentation	723
5.115.1.1	config_free	723
5.115.1.2	config_key_lookup	723
5.115.1.3	config_load_cmdline_settings	723
5.115.1.4	config_load_database_settings	724
5.115.1.5	config_load_defaults	724
5.115.1.6	config_load_file_settings	724
5.115.1.7	config_output_help	725
5.115.1.8	config_output_settings	725
5.115.1.9	config_output_value	725
5.115.1.10	config_output_value_generic	726
5.115.1.11	config_validate_settings	726
5.115.1.12	config_value_set	726
5.115.2	Variable Documentation	727
5.115.2.1	cmdline_config_data	727
5.115.2.2	exit_and_dump	727
5.115.2.3	magma	727
5.115.2.4	threadBuffer	728
5.116	src/engine/config/global/global.h File Reference	729
5.116.1	Function Documentation	730
5.116.1.1	config_fetch_host_number	730
5.116.1.2	config_fetch_settings	730
5.116.1.3	config_free	730
5.116.1.4	config_key_lookup	731
5.116.1.5	config_load_cmdline_settings	731
5.116.1.6	config_load_database_settings	731
5.116.1.7	config_load_defaults	732
5.116.1.8	config_load_file_settings	732
5.116.1.9	config_output_help	732
5.116.1.10	config_output_settings	733
5.116.1.11	config_output_value	733
5.116.1.12	config_output_value_generic	733
5.116.1.13	config_validate_settings	734
5.116.1.14	config_value_set	734
5.117	src/engine/config/relay/relay.c File Reference	735
5.117.1	Function Documentation	735

5.117.1.1 relay_alloc	735
5.117.1.2 relay_config	735
5.117.1.3 relay_counter	736
5.117.1.4 relay_free	736
5.117.1.5 relay_output_help	736
5.117.1.6 relay_output_settings	737
5.117.1.7 relay_set_value	737
5.117.1.8 relay_validate	737
5.118src/servers/smtp/relay.c File Reference	739
5.118.1 Function Documentation	739
5.118.1.1 smtp_client_close	739
5.118.1.2 smtp_client_connect	739
5.118.1.3 smtp_client_send_data	740
5.118.1.4 smtp_client_send_helo	740
5.118.1.5 smtp_client_send_mailfrom	740
5.118.1.6 smtp_client_send_nullfrom	741
5.118.1.7 smtp_client_send_rcptto	741
5.119src/engine/config/relay/relay.h File Reference	742
5.119.1 Function Documentation	742
5.119.1.1 relay_alloc	742
5.119.1.2 relay_config	742
5.119.1.3 relay_free	743
5.119.1.4 relay_output_help	743
5.119.1.5 relay_output_settings	744
5.119.1.6 relay_set_value	744
5.119.1.7 relay_validate	744
5.120src/engine/config/servers/servers.c File Reference	746
5.120.1 Function Documentation	747
5.120.1.1 servers_alloc	747
5.120.1.2 servers_config	747
5.120.1.3 servers_encryption_start	747
5.120.1.4 servers_encryption_stop	748
5.120.1.5 servers_free	748
5.120.1.6 servers_get_by_protocol	748
5.120.1.7 servers_get_by_socket	749
5.120.1.8 servers_get_count_using_port	749
5.120.1.9 servers_network_start	749
5.120.1.10servers_network_stop	749
5.120.1.11servers_output_help	750

5.120.1.12	servers_output_settings	750
5.120.1.13	servers_set_value	750
5.120.1.14	servers_validate	751
5.121	src/engine/config/servers/servers.h File Reference	752
5.121.1	Enumeration Type Documentation	753
5.121.1.1	M_PORT	753
5.121.1.2	M_PROTOCOL	753
5.121.2	Function Documentation	753
5.121.2.1	servers_alloc	753
5.121.2.2	servers_config	754
5.121.2.3	servers_encryption_start	754
5.121.2.4	servers_encryption_stop	754
5.121.2.5	servers_free	754
5.121.2.6	servers_get_by_protocol	755
5.121.2.7	servers_get_by_socket	755
5.121.2.8	servers_get_count_using_port	755
5.121.2.9	servers_network_start	756
5.121.2.10	servers_network_stop	756
5.121.2.11	servers_output_help	756
5.121.2.12	servers_output_settings	757
5.121.2.13	servers_set_value	757
5.121.2.14	servers_validate	757
5.122	src/servers/servers.h File Reference	759
5.122.1	Define Documentation	759
5.122.1.1	M_SSL_BIO_NOCLOSE	759
5.123	src/engine/context/args.c File Reference	760
5.123.1	Function Documentation	760
5.123.1.1	args_parse	760
5.123.1.2	display_usage	760
5.123.2	Variable Documentation	761
5.123.2.1	cmdline_config_data	761
5.123.2.2	exit_and_dump	761
5.124	src/engine/context/context.h File Reference	762
5.124.1	Function Documentation	763
5.124.1.1	args_parse	763
5.124.1.2	display_usage	763
5.124.1.3	process_start	763
5.124.1.4	process_stop	764
5.124.1.5	sanity_check	764

5.124.1.6	signal_segfault	765
5.124.1.7	signal_shutdown	765
5.124.1.8	signal_start	765
5.124.1.9	signal_status	766
5.124.1.10	signal_thread_start	766
5.124.1.11	system_change_root_directory	766
5.124.1.12	system_fork_daemon	767
5.124.1.13	system_init_core_dumps	767
5.124.1.14	system_init_impersonation	767
5.124.1.15	system_init_resource_limits	767
5.124.1.16	system_init_umask	768
5.124.1.17	system_ulimit_cur	768
5.124.1.18	system_ulimit_max	768
5.124.1.19	thread_start	769
5.124.1.20	thread_stop	769
5.125	src/engine/context/sanity.c File Reference	770
5.125.1	Function Documentation	770
5.125.1.1	sanity_check	770
5.126	src/engine/context/signal.c File Reference	771
5.126.1	Function Documentation	771
5.126.1.1	signal_refresh	771
5.126.1.2	signal_segfault	771
5.126.1.3	signal_shutdown	772
5.126.1.4	signal_start	772
5.126.1.5	signal_status	773
5.126.1.6	signal_thread_start	773
5.126.2	Variable Documentation	773
5.126.2.1	sig_hup_mutex	773
5.127	src/engine/context/system.c File Reference	774
5.127.1	Function Documentation	774
5.127.1.1	system_change_root_directory	774
5.127.1.2	system_fork_daemon	774
5.127.1.3	system_init_core_dumps	775
5.127.1.4	system_init_impersonation	775
5.127.1.5	system_init_resource_limits	775
5.127.1.6	system_init_umask	776
5.127.1.7	system_ulimit_cur	776
5.127.1.8	system_ulimit_max	776
5.128	src/engine/controller/controller.h File Reference	777

5.128.1 Function Documentation	777
5.128.1.1 dequeue	777
5.128.1.2 enqueue	777
5.128.1.3 protocol_init	778
5.128.1.4 protocol_process	778
5.128.1.5 queue_init	779
5.128.1.6 queue_shutdown	779
5.128.1.7 queue_signal	779
5.128.1.8 requeue	779
5.129src/engine/controller/protocol.c File Reference	781
5.129.1 Function Documentation	781
5.129.1.1 protocol_enqueue	781
5.129.1.2 protocol_init	781
5.129.1.3 protocol_process	782
5.129.1.4 protocol_secure	782
5.129.1.5 protocol_type	782
5.130src/engine/controller/queue.c File Reference	783
5.130.1 Function Documentation	783
5.130.1.1 dequeue	783
5.130.1.2 enqueue	784
5.130.1.3 queue_init	784
5.130.1.4 queue_shutdown	784
5.130.1.5 queue_signal	785
5.130.1.6 requeue	785
5.130.2 Variable Documentation	785
5.130.2.1 items	785
5.130.2.2 lock	785
5.130.2.3 queue	786
5.130.2.4 sema	786
5.130.2.5 workers	786
5.131src/engine/engine.h File Reference	787
5.132src/engine/log/log.c File Reference	788
5.132.1 Function Documentation	788
5.132.1.1 log_backtrace	788
5.132.1.2 log_disable	788
5.132.1.3 log_enable	789
5.132.1.4 log_internal	789
5.132.1.5 log_rotate	789
5.132.1.6 log_start	789

5.132.2 Variable Documentation	790
5.132.2.1 log_date	790
5.132.2.2 log_descriptor	790
5.132.2.3 log_enabled	790
5.132.2.4 log_mutex	790
5.133src/engine/log/log.h File Reference	791
5.133.1 Define Documentation	791
5.133.1.1 log_check	791
5.133.1.2 log_critical	792
5.133.1.3 log_error	792
5.133.1.4 log_info	792
5.133.1.5 log_options	792
5.133.1.6 log_pedantic	792
5.133.1.7 log_warn	792
5.133.1.8 MAGMA_LOG_LEVELS	792
5.133.2 Enumeration Type Documentation	792
5.133.2.1 M_LOG_OPTIONS	792
5.133.3 Function Documentation	793
5.133.3.1 log_backtrace	793
5.133.3.2 log_disable	793
5.133.3.3 log_enable	793
5.133.3.4 log_internal	794
5.133.3.5 log_rotate	794
5.133.3.6 log_start	794
5.134src/engine/status/build.c File Reference	795
5.134.1 Function Documentation	795
5.134.1.1 build_commit	795
5.134.1.2 build_stamp	795
5.134.1.3 build_version	796
5.134.1.4 build_version_major	796
5.134.1.5 build_version_minor	796
5.134.1.6 build_version_patch	796
5.135src/engine/status/performance.c File Reference	797
5.135.1 Function Documentation	797
5.135.1.1 perf_rdtsc	797
5.136src/engine/status/statistics.c File Reference	798
5.136.1 Function Documentation	799
5.136.1.1 stats_adjust_by_name	799
5.136.1.2 stats_adjust_by_num	799

5.136.1.3 stats_decrement_by_name	800
5.136.1.4 stats_decrement_by_num	800
5.136.1.5 stats_derived_count	800
5.136.1.6 stats_derived_name	800
5.136.1.7 stats_derived_value	801
5.136.1.8 stats_get_count	801
5.136.1.9 stats_get_name	801
5.136.1.10 stats_get_name_pos	802
5.136.1.11 stats_get_value_by_name	802
5.136.1.12 stats_get_value_by_num	802
5.136.1.13 stats_increment_by_name	802
5.136.1.14 stats_increment_by_num	803
5.136.1.15 stats_init	803
5.136.1.16 stats_set_by_name	803
5.136.1.17 stats_set_by_num	804
5.136.1.18 stats_shutdown	804
5.136.1.19 stats_sum_errors	804
5.136.2 Variable Documentation	804
5.136.2.1 count	804
5.136.2.2 derived	805
5.136.2.3 locks	805
5.136.2.4 names	805
5.136.2.5 stats	805
5.136.2.6 values	805
5.137 src/web/statistics/statistics.c File Reference	806
5.137.1 Function Documentation	806
5.137.1.1 statistics_process	806
5.137.2 Variable Documentation	806
5.137.2.1 portal_stats	806
5.138 src/engine/status/status.c File Reference	807
5.138.1 Function Documentation	807
5.138.1.1 status	807
5.138.1.2 status_get	808
5.138.1.3 status_pid	808
5.138.1.4 status_process	808
5.138.1.5 status_set	808
5.138.1.6 status_signal	809
5.138.1.7 status_startup	809
5.138.2 Variable Documentation	809

5.138.2.1 pid	809
5.138.2.2 process	809
5.138.2.3 startup	809
5.138.2.4 status_level	809
5.138.2.5 status_lock	809
5.139src/engine/status/status.h File Reference	810
5.139.1 Function Documentation	812
5.139.1.1 build_commit	812
5.139.1.2 build_stamp	812
5.139.1.3 build_version	812
5.139.1.4 build_version_major	812
5.139.1.5 build_version_minor	812
5.139.1.6 build_version_patch	813
5.139.1.7 perf_rdtsc	813
5.139.1.8 stats_adjust_by_name	813
5.139.1.9 stats_adjust_by_num	813
5.139.1.10stats_decrement_by_name	814
5.139.1.11stats_decrement_by_num	814
5.139.1.12stats_derived_count	814
5.139.1.13stats_derived_name	814
5.139.1.14stats_derived_value	815
5.139.1.15stats_get_count	815
5.139.1.16stats_get_name	815
5.139.1.17stats_get_name_pos	815
5.139.1.18stats_get_value_by_name	816
5.139.1.19stats_get_value_by_num	816
5.139.1.20stats_increment_by_name	816
5.139.1.21stats_increment_by_num	817
5.139.1.22stats_init	817
5.139.1.23stats_set_by_name	817
5.139.1.24stats_set_by_num	817
5.139.1.25stats_shutdown	818
5.139.1.26stats_sum_errors	818
5.139.1.27status	818
5.139.1.28status_get	818
5.139.1.29status_pid	819
5.139.1.30status_process	819
5.139.1.31status_set	819
5.139.1.32status_signal	819

5.139.1.33	status_startup	820
5.140	src/magma.c File Reference	821
5.140.1	Function Documentation	821
5.140.1.1	main	821
5.140.2	Variable Documentation	821
5.140.2.1	log_descriptor	821
5.141	src/magma.h File Reference	822
5.141.1	Define Documentation	823
5.141.1.1	buflen	823
5.141.1.2	bufptr	823
5.141.2	Variable Documentation	823
5.141.2.1	magma	823
5.141.2.2	threadBuffer	824
5.142	src/network/addresses.c File Reference	825
5.142.1	Function Documentation	825
5.142.1.1	con_addr	825
5.142.1.2	con_addr_octet	826
5.142.1.3	con_addr_presentation	826
5.142.1.4	con_addr_reversed	826
5.142.1.5	con_addr_segment	827
5.142.1.6	con_addr_standard	827
5.142.1.7	con_addr_subnet	827
5.142.1.8	con_addr_word	828
5.143	src/network/clients.c File Reference	829
5.143.1	Function Documentation	829
5.143.1.1	client_close	829
5.143.1.2	client_connect	829
5.143.1.3	client_secure	830
5.143.1.4	client_status	830
5.144	src/network/connections.c File Reference	831
5.144.1	Function Documentation	831
5.144.1.1	con_decrement_refs	831
5.144.1.2	con_destroy	832
5.144.1.3	con_flush	832
5.144.1.4	con_increment_refs	832
5.144.1.5	con_init	832
5.144.1.6	con_init_network_buffer	833
5.144.1.7	con_localhost	833
5.144.1.8	con_private	833

5.144.1.9 con_secure	834
5.144.1.10con_status	834
5.145src/network/dmtp.h File Reference	835
5.146src/providers/dime/signet-resolver/dmtp.h File Reference	836
5.146.1 Define Documentation	837
5.146.1.1 DMTP_LINE_BUF_SIZE	837
5.146.1.2 DMTP_MAX_MX_RETRIES	837
5.146.1.3 DMTP_PORT	837
5.146.1.4 DMTP_PORT_DUAL	837
5.146.1.5 DMTP_V1_CIPHER_LIST	838
5.146.2 Enumeration Type Documentation	838
5.146.2.1 dmtp_mail_datatype_t	838
5.146.2.2 dmtp_mail_rettype_t	838
5.146.2.3 dmtp_mode_t	838
5.146.3 Function Documentation	838
5.146.3.1 _sgnt_resolv_dmtp_expect_banner	838
5.146.3.2 _sgnt_resolv_dmtp_initiate_starttls	839
5.146.3.3 _sgnt_resolv_dmtp_issue_command	839
5.146.3.4 _sgnt_resolv_dmtp_send_and_read	839
5.146.3.5 _sgnt_resolv_dmtp_str_to_mode	840
5.146.3.6 _sgnt_resolv_dmtp_write_data	840
5.146.3.7 _sgnt_resolv_parse_line_code	841
5.146.3.8 _sgnt_resolv_read_dmtp_line	841
5.146.3.9 _sgnt_resolv_read_dmtp_multiline	841
5.146.3.10PUBLIC_FUNC_DECL	843
5.146.3.11PUBLIC_FUNC_DECL	843
5.146.3.12PUBLIC_FUNC_DECL	843
5.146.3.13PUBLIC_FUNC_DECL	843
5.146.3.14PUBLIC_FUNC_DECL	843
5.146.3.15PUBLIC_FUNC_DECL	843
5.146.3.16PUBLIC_FUNC_DECL	843
5.146.3.17PUBLIC_FUNC_DECL	843
5.146.3.18PUBLIC_FUNC_DECL	843
5.146.3.19PUBLIC_FUNC_DECL	843
5.146.3.20PUBLIC_FUNC_DECL	843
5.146.3.21PUBLIC_FUNC_DECL	843
5.146.3.22PUBLIC_FUNC_DECL	843
5.146.3.23PUBLIC_FUNC_DECL	843
5.146.3.24PUBLIC_FUNC_DECL	843

5.146.3.25PUBLIC_FUNC_DECL	843
5.146.3.26PUBLIC_FUNC_DECL	843
5.146.3.27PUBLIC_FUNC_DECL	843
5.146.3.28PUBLIC_FUNC_DECL	843
5.146.3.29PUBLIC_FUNC_DECL	843
5.147src/servers/dmtp/dmtp.h File Reference	844
5.147.1 Function Documentation	845
5.147.1.1 dmtp_compare	845
5.147.1.2 dmtp_data	845
5.147.1.3 dmtp_ehlo	845
5.147.1.4 dmtp_helo	845
5.147.1.5 dmtp_help	846
5.147.1.6 dmtp_hist	846
5.147.1.7 dmtp_init	846
5.147.1.8 dmtp_invalid	846
5.147.1.9 dmtp_mail	846
5.147.1.10dmtp_mode	847
5.147.1.11dmtp_noop	847
5.147.1.12dmtp_process	847
5.147.1.13dmtp_quit	847
5.147.1.14dmtp_rcpt	848
5.147.1.15dmtp_requeue	848
5.147.1.16dmtp_rset	848
5.147.1.17dmtp_session_destroy	848
5.147.1.18dmtp_session_reset	849
5.147.1.19dmtp_sgmt	849
5.147.1.20dmtp_sort	849
5.147.1.21dmtp_verb	849
5.147.1.22dmtp_vrfy	849
5.148src/network/http.h File Reference	850
5.148.1 Enumeration Type Documentation	850
5.148.1.1 "@49	850
5.148.1.2 HTTP_DATA	850
5.148.1.3 http_method_t	850
5.149src/servers/http/http.h File Reference	852
5.149.1 Enumeration Type Documentation	855
5.149.1.1 "@83	855
5.149.1.2 "@84	855
5.149.1.3 "@85	855

5.149.2 Function Documentation	856
5.149.2.1 get_header_opt	856
5.149.2.2 get_header_value_noopt	856
5.149.2.3 http_body	856
5.149.2.4 http_close	857
5.149.2.5 http_content_load_directory	857
5.149.2.6 http_content_load_fonts	857
5.149.2.7 http_content_refresh	858
5.149.2.8 http_content_start	858
5.149.2.9 http_content_stop	858
5.149.2.10 http_data_free	858
5.149.2.11 http_data_get	859
5.149.2.12 http_data_header_parse	859
5.149.2.13 http_data_header_parse_line	859
5.149.2.14 http_data_value_decode	860
5.149.2.15 http_data_value_parse	860
5.149.2.16 http_free_content	861
5.149.2.17 http_get_static	861
5.149.2.18 http_get_template	861
5.149.2.19 http_init	861
5.149.2.20 http_load_file	862
5.149.2.21 http_page_free	862
5.149.2.22 http_page_get	863
5.149.2.23 http_parse_context	863
5.149.2.24 http_parse_header	863
5.149.2.25 http_parse_method	864
5.149.2.26 http_parse_origin	864
5.149.2.27 http_parse_pairs	865
5.149.2.28 http_print_301	865
5.149.2.29 http_print_400	865
5.149.2.30 http_print_403	866
5.149.2.31 http_print_404	866
5.149.2.32 http_print_405	866
5.149.2.33 http_print_500	867
5.149.2.34 http_print_500_log	867
5.149.2.35 http_print_501	867
5.149.2.36 http_process	868
5.149.2.37 http_requeue	868
5.149.2.38 http_response	868

5.149.2.39	http_response_allow_cross	869
5.149.2.40	http_response_connection	869
5.149.2.41	http_response_cookie	869
5.149.2.42	http_response_header	869
5.149.2.43	http_response_options	870
5.149.2.44	http_response_status	870
5.149.2.45	http_session_destroy	871
5.149.2.46	http_session_reset	871
5.149.2.47	multipart_get_boundary	871
5.150	src/network/imap.h File Reference	872
5.150.1	Typedef Documentation	872
5.150.1.1	imap_arguments_t	872
5.150.2	Function Documentation	872
5.150.2.1	__attribute__	872
5.150.3	Variable Documentation	872
5.150.3.1	imap_session_t	872
5.151	src/servers/imap/imap.h File Reference	873
5.151.1	Define Documentation	877
5.151.1.1	IMAP_ARGUMENT_TYPE_ARRAY	877
5.151.1.2	IMAP_ARGUMENT_TYPE_ASTRING	877
5.151.1.3	IMAP_ARGUMENT_TYPE_EMPTY	877
5.151.1.4	IMAP_ARGUMENT_TYPE_LITERAL	877
5.151.1.5	IMAP_ARGUMENT_TYPE_NSTRING	877
5.151.1.6	IMAP_ARGUMENT_TYPE_QSTRING	877
5.151.1.7	IMAP_ARRAY_RECURSION_LIMIT	877
5.151.1.8	IMAP_FETCH_BODY_HEADER	878
5.151.1.9	IMAP_FETCH_BODY_HEADER_FIELDS	878
5.151.1.10	IMAP_FETCH_BODY_HEADER_FIELDS_NOT	878
5.151.1.11	IMAP_FETCH_BODY_MIME	878
5.151.1.12	IMAP_FETCH_BODY_PART	878
5.151.1.13	IMAP_FETCH_BODY_TEXT	878
5.151.1.14	IMAP_FLAG_ADD	878
5.151.1.15	IMAP_FLAG_REMOVE	878
5.151.1.16	IMAP_FLAG_REPLACE	878
5.151.1.17	IMAP_FLAG_SILENT	878
5.151.1.18	IMAP_FOLDER_RECURSION_LIMIT	879
5.151.1.19	IMAP_SEARCH_RECURSION_LIMIT	879
5.151.2	Function Documentation	879
5.151.2.1	imap_append	879

5.151.2.2	imap_append_message	879
5.151.2.3	imap_build_array	879
5.151.2.4	imap_build_array_isliteral	879
5.151.2.5	imap_capability	880
5.151.2.6	imap_check	880
5.151.2.7	imap_close	880
5.151.2.8	imap_command_parser	880
5.151.2.9	imap_compare	881
5.151.2.10	imap_copy	881
5.151.2.11	imap_count_folder_levels	881
5.151.2.12	imap_create	881
5.151.2.13	imap_delete	882
5.151.2.14	imap_duplicate_messages	882
5.151.2.15	imap_examine	883
5.151.2.16	imap_expunge	883
5.151.2.17	imap_fetch	883
5.151.2.18	imap_fetch_body	883
5.151.2.19	imap_fetch_body_header	883
5.151.2.20	imap_fetch_body_mime	884
5.151.2.21	imap_fetch_body_part	884
5.151.2.22	imap_fetch_body_portion	884
5.151.2.23	imap_fetch_body_tag	884
5.151.2.24	imap_fetch_bodystructure	884
5.151.2.25	imap_fetch_envelope	884
5.151.2.26	imap_fetch_free_items	884
5.151.2.27	imap_fetch_message	885
5.151.2.28	imap_fetch_parse_partial	885
5.151.2.29	imap_fetch_response_add	885
5.151.2.30	imap_fetch_response_free	885
5.151.2.31	imap_fetch_return_header	885
5.151.2.32	imap_fetch_return_message	886
5.151.2.33	imap_fetch_return_mime	886
5.151.2.34	imap_fetch_return_text	886
5.151.2.35	imap_flag_action	886
5.151.2.36	imap_flag_parse	886
5.151.2.37	imap_folder_compare	886
5.151.2.38	imap_folder_create	886
5.151.2.39	imap_folder_name_escaped	887
5.151.2.40	imap_folder_remove	887

5.151.2.4	imap_folder_rename	888
5.151.2.42	imap_folder_status	888
5.151.2.43	imap_get_ar_ar	889
5.151.2.44	imap_get_flag	889
5.151.2.45	imap_get_ptr	889
5.151.2.46	imap_get_st_ar	890
5.151.2.47	imap_get_type_ar	890
5.151.2.48	imap_id	890
5.151.2.49	imap_idle	890
5.151.2.50	imap_init	891
5.151.2.5	imap_invalid	891
5.151.2.52	imap_list	891
5.151.2.53	imap_login	891
5.151.2.54	imap_logout	892
5.151.2.55	imap_lsub	892
5.151.2.56	imap_message_copier	892
5.151.2.57	imap_message_expunge	892
5.151.2.58	imap_narrow_folders	892
5.151.2.59	imap_narrow_messages	893
5.151.2.60	imap_next_folder_order	893
5.151.2.6	imap_noop	893
5.151.2.62	imap_parse_address	893
5.151.2.63	imap_parse_address_breaker	893
5.151.2.64	imap_parse_address_part	894
5.151.2.65	imap_parse_address_put	894
5.151.2.66	imap_parse_arguments	894
5.151.2.67	imap_parse_array	894
5.151.2.68	imap_parse_astring	895
5.151.2.69	imap_parse_dataitems	895
5.151.2.70	imap_parse_literal	896
5.151.2.7	imap_parse_nstring	896
5.151.2.72	imap_parse_qstring	896
5.151.2.73	imap_process	897
5.151.2.74	imap_range_build	897
5.151.2.75	imap_rename	897
5.151.2.76	imap_requeue	898
5.151.2.77	imap_search	898
5.151.2.78	imap_search_flag	898
5.151.2.79	imap_search_messages	898

5.151.2.80imap_search_messages_body	898
5.151.2.81imap_search_messages_date	898
5.151.2.82imap_search_messages_date_compare	899
5.151.2.83imap_search_messages_header	899
5.151.2.84imap_search_messages_inner	899
5.151.2.85imap_search_messages_range	899
5.151.2.86imap_search_messages_size	899
5.151.2.87imap_search_messages_text	899
5.151.2.88imap_select	900
5.151.2.89imap_session_destroy	900
5.151.2.90imap_session_update	900
5.151.2.91imap_sort	900
5.151.2.92imap_starttls	900
5.151.2.93imap_status	901
5.151.2.94imap_store	901
5.151.2.95imap_subscribe	901
5.151.2.96imap_unsubscribe	901
5.151.2.97imap_update_flags	901
5.151.2.98imap_valid_folder_name	902
5.151.2.99imap_valid_sequence	902
5.152src/network/listeners.c File Reference	903
5.152.1 Function Documentation	903
5.152.1.1 net_accept	903
5.152.1.2 net_init	903
5.152.1.3 net_listen	903
5.152.1.4 net_shutdown	903
5.152.1.5 net_trigger	904
5.153src/network/meta.h File Reference	905
5.153.1 Enumeration Type Documentation	905
5.153.1.1 "@54	905
5.153.1.2 META_CHECKPOINT	905
5.153.1.3 META_GET	906
5.153.1.4 META_LOCK_STATUS	906
5.153.1.5 META_PROTOCOL	906
5.153.1.6 META_USER_FLAGS	906
5.153.2 Function Documentation	907
5.153.2.1 __attribute__	907
5.153.3 Variable Documentation	907
5.153.3.1 meta_alert_t	907

5.153.3.2 meta_alias_t	907
5.153.3.3 meta_folder_t	907
5.153.3.4 meta_message_t	907
5.153.3.5 meta_user_t	907
5.154src/objects/meta/meta.h File Reference	908
5.154.1 Function Documentation	911
5.154.1.1 alert_alloc	911
5.154.1.2 alias_alloc	911
5.154.1.3 meta_alloc	912
5.154.1.4 meta_crypto_keys_create	912
5.154.1.5 meta_data_acknowledge_alert	912
5.154.1.6 meta_data_delete_folder	913
5.154.1.7 meta_data_delete_tag	913
5.154.1.8 meta_data_fetch_alerts	913
5.154.1.9 meta_data_fetch_all_tags	914
5.154.1.10meta_data_fetch_folders	914
5.154.1.11meta_data_fetch_keys	914
5.154.1.12meta_data_fetch_mailbox_aliases	915
5.154.1.13meta_data_fetch_shard	915
5.154.1.14meta_data_fetch_user	915
5.154.1.15meta_data_flags_add	916
5.154.1.16meta_data_flags_remove	916
5.154.1.17meta_data_flags_replace	916
5.154.1.18meta_data_insert_folder	917
5.154.1.19meta_data_insert_keys	917
5.154.1.20meta_data_insert_shard	918
5.154.1.21meta_data_insert_tag	918
5.154.1.22meta_data_truncate_tags	918
5.154.1.23meta_data_update_folder_name	919
5.154.1.24meta_data_update_lock	919
5.154.1.25meta_data_update_log	919
5.154.1.26meta_folder_stats_tag_alloc	919
5.154.1.27meta_folders_by_name	920
5.154.1.28meta_folders_by_number	920
5.154.1.29meta_folders_children	920
5.154.1.30meta_folders_name	921
5.154.1.31meta_folders_stats_tags	921
5.154.1.32meta_free	922
5.154.1.33meta_get	922

5.154.1.34	meta_inx_find	922
5.154.1.35	meta_inx_remove	923
5.154.1.36	meta_update_aliases	923
5.154.1.37	meta_update_contacts	923
5.154.1.38	meta_update_folders	924
5.154.1.39	meta_update_keys	924
5.154.1.40	meta_update_message_folders	925
5.154.1.41	meta_update_realms	925
5.154.1.42	meta_update_user	925
5.154.1.43	meta_user_ref_add	926
5.154.1.44	meta_user_ref_dec	926
5.154.1.45	meta_user_ref_protocol_total	927
5.154.1.46	meta_user_ref_stamp	927
5.154.1.47	meta_user_ref_total	927
5.154.1.48	meta_user_rlock	928
5.154.1.49	meta_user_serial_check	928
5.154.1.50	meta_user_serial_get	928
5.154.1.51	meta_user_serial_set	929
5.154.1.52	meta_user_unlock	929
5.154.1.53	meta_user_wlock	929
5.155	src/network/network.h File Reference	931
5.155.1	Enumeration Type Documentation	934
5.155.1.1	"@55	934
5.155.2	Function Documentation	934
5.155.2.1	__attribute__	934
5.155.2.2	client_close	934
5.155.2.3	client_connect	935
5.155.2.4	client_print	935
5.155.2.5	client_read	935
5.155.2.6	client_read_line	936
5.155.2.7	client_secure	936
5.155.2.8	client_status	936
5.155.2.9	client_write	936
5.155.2.10	con_addr	937
5.155.2.11	con_addr_octet	937
5.155.2.12	con_addr_presentation	937
5.155.2.13	con_addr_reversed	938
5.155.2.14	con_addr_segment	938
5.155.2.15	con_addr_standard	938

5.155.2.16	con_addr_subnet	939
5.155.2.17	con_addr_word	939
5.155.2.18	con_decrement_refs	939
5.155.2.19	con_destroy	940
5.155.2.20	con_flush	940
5.155.2.21	con_increment_refs	940
5.155.2.22	con_init	940
5.155.2.23	con_init_network_buffer	941
5.155.2.24	con_localhost	941
5.155.2.25	con_print	941
5.155.2.26	con_private	942
5.155.2.27	con_read	942
5.155.2.28	con_read_line	942
5.155.2.29	con_reverse_check	943
5.155.2.30	con_reverse_domain	943
5.155.2.31	con_reverse_enqueue	944
5.155.2.32	con_reverse_lookup	944
5.155.2.33	con_reverse_status	944
5.155.2.34	con_secure	944
5.155.2.35	con_status	945
5.155.2.36	con_write_bl	945
5.155.2.37	con_write_ns	946
5.155.2.38	con_write_pl	946
5.155.2.39	con_write_st	946
5.155.2.40	net_init	947
5.155.2.41	net_listen	947
5.155.2.42	net_set_blocking	947
5.155.2.43	net_set_buffer_length	947
5.155.2.44	net_set_keeppalive	948
5.155.2.45	net_set_linger	948
5.155.2.46	net_set_nodelay	948
5.155.2.47	net_set_reuseable_address	949
5.155.2.48	net_set_timeout	949
5.155.2.49	net_shutdown	949
5.155.2.50	net_trigger	950
5.155.2.51	protocol_type	950
5.155.3	Variable Documentation	950
5.155.3.1	client_t	950
5.155.3.2	command_t	950

5.155.3.3 connection_t	950
5.156src/providers/dime/common/network.h File Reference	951
5.156.1 Function Documentation	951
5.156.1.1 _connect_timeout	951
5.156.1.2 PUBLIC_FUNC_DECL	951
5.157src/network/options.c File Reference	952
5.157.1 Function Documentation	952
5.157.1.1 net_set_blocking	952
5.157.1.2 net_set_buffer_length	952
5.157.1.3 net_set_keepalive	953
5.157.1.4 net_set_linger	953
5.157.1.5 net_set_nodelay	954
5.157.1.6 net_set_reuseable_address	954
5.157.1.7 net_set_timeout	954
5.158src/network/pop.h File Reference	955
5.159src/servers/pop/pop.h File Reference	956
5.159.1 Function Documentation	957
5.159.1.1 pop_capa	957
5.159.1.2 pop_compare	958
5.159.1.3 pop_dele	958
5.159.1.4 pop_get_last	958
5.159.1.5 pop_get_message	958
5.159.1.6 pop_init	959
5.159.1.7 pop_invalid	959
5.159.1.8 pop_last	959
5.159.1.9 pop_list	960
5.159.1.10pop_noop	960
5.159.1.11pop_num_parse	960
5.159.1.12pop_pass	961
5.159.1.13pop_pass_parse	961
5.159.1.14pop_process	962
5.159.1.15pop_quit	962
5.159.1.16pop_requeue	962
5.159.1.17pop_retr	962
5.159.1.18pop_rset	962
5.159.1.19pop_session_destroy	963
5.159.1.20pop_session_reset	963
5.159.1.21pop_sort	963
5.159.1.22pop_starttls	964

5.159.1.23	pop_stat	964
5.159.1.24	pop_top	964
5.159.1.25	pop_top_parse	965
5.159.1.26	pop_total_messages	965
5.159.1.27	pop_total_size	965
5.159.1.28	pop_uidl	966
5.159.1.29	pop_user	966
5.159.1.30	pop_user_parse	966
5.160	src/network/read.c File Reference	967
5.160.1	Function Documentation	967
5.160.1.1	client_read	967
5.160.1.2	client_read_line	967
5.160.1.3	con_read	968
5.160.1.4	con_read_line	968
5.161	src/network/reverse.c File Reference	969
5.161.1	Function Documentation	969
5.161.1.1	con_reverse_check	969
5.161.1.2	con_reverse_domain	969
5.161.1.3	con_reverse_enqueue	970
5.161.1.4	con_reverse_lookup	970
5.161.1.5	con_reverse_status	970
5.162	src/network/sessions.h File Reference	971
5.162.1	Enumeration Type Documentation	971
5.162.1.1	"@56	971
5.162.2	Function Documentation	971
5.162.2.1	__attribute__	971
5.162.3	Variable Documentation	971
5.162.3.1	attachment_t	971
5.162.3.2	composition_t	971
5.162.3.3	session_t	972
5.163	src/objects/sessions/sessions.h File Reference	973
5.163.1	Function Documentation	974
5.163.1.1	sess_create	974
5.163.1.2	sess_destroy	974
5.163.1.3	sess_get	975
5.163.1.4	sess_key	975
5.163.1.5	sess_number	975
5.163.1.6	sess_ref_add	975
5.163.1.7	sess_ref_dec	976

5.163.1.8	sess_ref_stamp	976
5.163.1.9	sess_ref_total	976
5.163.1.10	sess_refresh_check	977
5.163.1.11	sess_refresh_flush	977
5.163.1.12	sess_refresh_stamp	977
5.163.1.13	sess_release	978
5.163.1.14	sess_release_attachment	978
5.163.1.15	sess_release_composition	978
5.163.1.16	sess_serial_check	979
5.163.1.17	sess_token	979
5.163.1.18	sess_trigger	979
5.163.1.19	sess_update	979
5.164	src/network/smtp.h File Reference	981
5.164.1	Enumeration Type Documentation	981
5.164.1.1	"@57	981
5.165	src/servers/smtp/smtp.h File Reference	983
5.165.1	Function Documentation	986
5.165.1.1	smtp_accept_message	986
5.165.1.2	smtp_add_bypass_entry	986
5.165.1.3	smtp_add_inbound	987
5.165.1.4	smtp_add_outbound	987
5.165.1.5	smtp_add_recipient	987
5.165.1.6	smtp_auth_login	988
5.165.1.7	smtp_auth_plain	988
5.165.1.8	smtp_bounce	988
5.165.1.9	smtp_bypass_check	988
5.165.1.10	smtp_check_authorized_from	989
5.165.1.11	smtp_check_duplicate_recipient	989
5.165.1.12	smtp_check_filters	989
5.165.1.13	smtp_check_greylist	990
5.165.1.14	smtp_check_rbl	990
5.165.1.15	smtp_check_receive_quota	991
5.165.1.16	smtp_check_transmit_quota	991
5.165.1.17	smtp_client_close	991
5.165.1.18	smtp_client_connect	992
5.165.1.19	smtp_client_send_data	992
5.165.1.20	smtp_client_send_helo	992
5.165.1.21	smtp_client_send_mailfrom	993
5.165.1.22	smtp_client_send_nullfrom	993

5.165.1.23smtp_client_send_rcptto	994
5.165.1.24smtp_compare	994
5.165.1.25smtp_data	994
5.165.1.26smtp_disabled	994
5.165.1.27smtp_ehlo	994
5.165.1.28smtp_fetch_authorization	995
5.165.1.29smtp_fetch_autoreply	995
5.165.1.30smtp_fetch_inbound	996
5.165.1.31smtp_fetch_rollmessages	997
5.165.1.32smtp_forward_message	997
5.165.1.33smtp_free_inbound	997
5.165.1.34smtp_free_outbound	998
5.165.1.35smtp_free_recipients	998
5.165.1.36smtp_get_action	998
5.165.1.37smtp_helo	999
5.165.1.38smtp_init	999
5.165.1.39smtp_insert_spamsig	999
5.165.1.40smtp_invalid	1000
5.165.1.41smtp_list_free_filter	1000
5.165.1.42smtp_mail_from	1000
5.165.1.43smtp_noop	1001
5.165.1.44smtp_parse_auth	1001
5.165.1.45smtp_parse_helo_domain	1001
5.165.1.46smtp_parse_mail_from_path	1002
5.165.1.47smtp_parse_rcpt_to	1002
5.165.1.48smtp_process	1002
5.165.1.49smtp_quit	1003
5.165.1.50smtp_rcpt_to	1003
5.165.1.51smtp_relay_message	1003
5.165.1.52smtp_reply	1004
5.165.1.53smtp_requeue	1004
5.165.1.54smtp_rollout	1004
5.165.1.55smtp_rset	1005
5.165.1.56smtp_send_message	1005
5.165.1.57smtp_session_destroy	1005
5.165.1.58smtp_session_reset	1006
5.165.1.59smtp_sort	1006
5.165.1.60smtp_starttls	1006
5.165.1.61smtp_store_message	1006

5.165.1.62smtp_store_spamsig	1007
5.165.1.63smtp_update_receive_stats	1007
5.165.1.64smtp_update_transmission_stats	1007
5.165.1.65submission_init	1008
5.166src/network/write.c File Reference	1009
5.166.1 Function Documentation	1009
5.166.1.1 client_print	1009
5.166.1.2 client_write	1010
5.166.1.3 con_print	1010
5.166.1.4 con_write_bl	1010
5.166.1.5 con_write_ns	1011
5.166.1.6 con_write_pl	1011
5.166.1.7 con_write_st	1012
5.167src/objects/auth/auth.c File Reference	1013
5.167.1 Function Documentation	1013
5.167.1.1 auth_alloc	1013
5.167.1.2 auth_challenge	1013
5.167.1.3 auth_free	1014
5.167.1.4 auth_login	1014
5.167.1.5 auth_response	1015
5.168src/objects/auth/auth.h File Reference	1016
5.168.1 Enumeration Type Documentation	1017
5.168.1.1 auth_lock_status_t	1017
5.168.2 Function Documentation	1017
5.168.2.1 auth_alloc	1017
5.168.2.2 auth_challenge	1017
5.168.2.3 auth_data_fetch	1018
5.168.2.4 auth_data_update_legacy	1018
5.168.2.5 auth_data_update_lock	1019
5.168.2.6 auth_free	1019
5.168.2.7 auth_legacy	1019
5.168.2.8 auth_legacy_alloc	1020
5.168.2.9 auth_legacy_free	1020
5.168.2.10auth_login	1020
5.168.2.11auth_response	1020
5.168.2.12auth_sanitize_address	1021
5.168.2.13auth_sanitize_username	1021
5.168.2.14auth_stacie	1021
5.168.2.15auth_stacie_alloc	1022

5.168.2.16	auth_stacie_cleanup	1022
5.168.2.17	auth_stacie_free	1023
5.169	src/objects/auth/legacy.c File Reference	1024
5.169.1	Function Documentation	1024
5.169.1.1	auth_legacy	1024
5.169.1.2	auth_legacy_alloc	1024
5.169.1.3	auth_legacy_free	1024
5.170	src/objects/auth/stacie.c File Reference	1025
5.170.1	Function Documentation	1025
5.170.1.1	auth_stacie	1025
5.170.1.2	auth_stacie_alloc	1026
5.170.1.3	auth_stacie_cleanup	1026
5.170.1.4	auth_stacie_free	1026
5.171	src/objects/auth/username.c File Reference	1028
5.171.1	Function Documentation	1028
5.171.1.1	auth_sanitize_address	1028
5.171.1.2	auth_sanitize_username	1028
5.172	src/objects/config/config.c File Reference	1029
5.172.1	Function Documentation	1029
5.172.1.1	user_config_alloc	1029
5.172.1.2	user_config_create	1029
5.172.1.3	user_config_edit	1030
5.172.1.4	user_config_entry_alloc	1030
5.172.1.5	user_config_entry_free	1030
5.172.1.6	user_config_free	1031
5.172.1.7	user_config_update	1031
5.173	src/web/portal/config.c File Reference	1032
5.173.1	Function Documentation	1032
5.173.1.1	portal_config_collection	1032
5.173.1.2	portal_config_entry	1032
5.173.1.3	portal_config_entry_flags	1032
5.174	src/objects/contacts/contacts.c File Reference	1034
5.174.1	Function Documentation	1034
5.174.1.1	contact_alloc	1034
5.174.1.2	contact_create	1035
5.174.1.3	contact_detail_alloc	1035
5.174.1.4	contact_detail_free	1035
5.174.1.5	contact_edit	1036
5.174.1.6	contact_free	1036

5.174.1.7	contact_move	1037
5.174.1.8	contact_name	1037
5.174.1.9	contact_remove	1037
5.174.1.10	contact_validate_detail	1038
5.174.1.11	contact_validate_name	1038
5.174.1.12	contacts_update	1038
5.175	src/objects/folders/contacts.c File Reference	1040
5.175.1	Function Documentation	1040
5.175.1.1	contact_folder_alloc	1040
5.175.1.2	contact_folder_create	1040
5.175.1.3	contact_folder_free	1041
5.175.1.4	contact_folder_remove	1041
5.175.1.5	contact_folder_rename	1042
5.176	src/web/portal/contacts.c File Reference	1043
5.176.1	Function Documentation	1043
5.176.1.1	portal_contact_detail_flags	1043
5.176.1.2	portal_contact_details	1043
5.177	src/objects/contacts/contacts.h File Reference	1044
5.177.1	Enumeration Type Documentation	1045
5.177.1.1	"@67	1045
5.177.2	Function Documentation	1045
5.177.2.1	__attribute__	1045
5.177.2.2	contact_alloc	1046
5.177.2.3	contact_create	1046
5.177.2.4	contact_delete	1046
5.177.2.5	contact_detail_alloc	1047
5.177.2.6	contact_detail_delete	1047
5.177.2.7	contact_detail_free	1047
5.177.2.8	contact_detail_upsert	1048
5.177.2.9	contact_details_fetch	1048
5.177.2.10	contact_edit	1048
5.177.2.11	contact_find_detail	1049
5.177.2.12	contact_find_name	1049
5.177.2.13	contact_find_number	1049
5.177.2.14	contact_free	1050
5.177.2.15	contact_insert	1050
5.177.2.16	contact_move	1050
5.177.2.17	contact_name	1051
5.177.2.18	contact_remove	1051

5.177.2.19	contact_update	1051
5.177.2.20	contact_update_stamp	1052
5.177.2.21	contact_validate_detail	1052
5.177.2.22	contact_validate_name	1052
5.177.2.23	contacts_fetch	1053
5.177.2.24	contacts_update	1053
5.177.3	Variable Documentation	1053
5.177.3.1	contact_detail_t	1053
5.177.3.2	contact_t	1054
5.178	src/objects/contacts/find.c File Reference	1055
5.178.1	Function Documentation	1055
5.178.1.1	contact_find_detail	1055
5.178.1.2	contact_find_name	1055
5.178.1.3	contact_find_number	1055
5.179	src/objects/folders/find.c File Reference	1057
5.179.1	Function Documentation	1057
5.179.1.1	magma_folder_find_full_name	1057
5.179.1.2	magma_folder_find_name	1057
5.179.1.3	magma_folder_find_number	1058
5.180	src/objects/folders/folders.c File Reference	1059
5.180.1	Function Documentation	1059
5.180.1.1	magma_folder_alloc	1059
5.180.1.2	magma_folder_children	1059
5.180.1.3	magma_folder_free	1060
5.180.1.4	magma_folder_funcs	1060
5.180.1.5	magma_folder_name	1060
5.181	src/objects/meta/folders.c File Reference	1062
5.181.1	Function Documentation	1062
5.181.1.1	meta_folder_stats_tag_alloc	1062
5.181.1.2	meta_folders_by_name	1062
5.181.1.3	meta_folders_by_number	1063
5.181.1.4	meta_folders_children	1063
5.181.1.5	meta_folders_name	1063
5.181.1.6	meta_folders_stats_tags	1064
5.182	src/servers/imap/folders.c File Reference	1065
5.182.1	Function Documentation	1065
5.182.1.1	imap_count_folder_levels	1065
5.182.1.2	imap_folder_compare	1066
5.182.1.3	imap_folder_create	1066

5.182.1.4	imap_folder_name_escaped	1066
5.182.1.5	imap_folder_remove	1067
5.182.1.6	imap_folder_rename	1067
5.182.1.7	imap_folder_status	1068
5.182.1.8	imap_narrow_folders	1068
5.182.1.9	imap_next_folder_order	1068
5.182.1.10	imap_valid_folder_name	1069
5.183	src/web/portal/folders.c File Reference	1070
5.183.1	Function Documentation	1070
5.183.1.1	portal_folder_contacts_add	1070
5.183.1.2	portal_folder_contacts_remove	1070
5.183.1.3	portal_folder_mail_add	1071
5.183.1.4	portal_folder_mail_remove	1071
5.184	src/objects/folders/folders.h File Reference	1073
5.184.1	Define Documentation	1074
5.184.1.1	FOLDER_LENGTH_LIMIT	1074
5.184.1.2	FOLDER_RECURSION_LIMIT	1074
5.184.2	Typedef Documentation	1074
5.184.2.1	contact_folder_t	1074
5.184.2.2	message_folder_t	1075
5.184.3	Enumeration Type Documentation	1075
5.184.3.1	"@68	1075
5.184.4	Function Documentation	1075
5.184.4.1	__attribute__	1075
5.184.4.2	contact_folder_alloc	1075
5.184.4.3	contact_folder_create	1075
5.184.4.4	contact_folder_free	1076
5.184.4.5	contact_folder_remove	1076
5.184.4.6	contact_folder_rename	1077
5.184.4.7	magma_folder_alloc	1077
5.184.4.8	magma_folder_children	1077
5.184.4.9	magma_folder_delete	1078
5.184.4.10	magma_folder_fetch	1078
5.184.4.11	magma_folder_find_full_name	1078
5.184.4.12	magma_folder_find_name	1079
5.184.4.13	magma_folder_find_number	1079
5.184.4.14	magma_folder_free	1079
5.184.4.15	magma_folder_funcs	1080
5.184.4.16	magma_folder_insert	1080

5.184.4.17	magma_folder_name	1080
5.184.4.18	magma_folder_rename	1081
5.184.4.19	message_folder_alloc	1081
5.184.4.20	message_folder_create	1082
5.184.4.21	message_folder_free	1082
5.184.4.22	message_folder_remove	1082
5.184.5	Variable Documentation	1082
5.184.5.1	magma_folder_t	1082
5.185	src/objects/folders/messages.c File Reference	1084
5.185.1	Function Documentation	1084
5.185.1.1	message_folder_alloc	1084
5.185.1.2	message_folder_create	1084
5.185.1.3	message_folder_free	1085
5.185.1.4	message_folder_remove	1085
5.186	src/objects/messages/messages.c File Reference	1086
5.186.1	Function Documentation	1086
5.186.1.1	message_alloc	1086
5.186.1.2	message_free	1086
5.186.1.3	messages_update	1087
5.187	src/providers/prime/messages/messages.c File Reference	1088
5.187.1	Function Documentation	1088
5.187.1.1	encrypted_message_alloc	1088
5.187.1.2	encrypted_message_cleanup	1088
5.187.1.3	encrypted_message_free	1088
5.187.1.4	naked_message_get	1088
5.187.1.5	naked_message_set	1089
5.188	src/servers/imap/messages.c File Reference	1090
5.188.1	Function Documentation	1090
5.188.1.1	imap_append_message	1090
5.188.1.2	imap_message_copier	1090
5.188.1.3	imap_message_expunge	1090
5.189	src/web/portal/messages.c File Reference	1091
5.189.1	Function Documentation	1091
5.189.1.1	portal_message_attachments	1091
5.189.1.2	portal_message_body	1091
5.189.1.3	portal_message_header	1092
5.189.1.4	portal_message_info	1092
5.189.1.5	portal_message_meta	1092
5.189.1.6	portal_message_security	1092

5.189.1.7 portal_message_server	1093
5.189.1.8 portal_message_source	1093
5.190src/objects/locks.c File Reference	1094
5.190.1 Define Documentation	1094
5.190.1.1 MAGMA_LOCK_EXPIRATION	1094
5.190.1.2 MAGMA_LOCK_TIMEOUT	1094
5.190.2 Function Documentation	1094
5.190.2.1 lock_get	1094
5.190.2.2 lock_release	1095
5.190.2.3 user_lock	1095
5.190.2.4 user_unlock	1096
5.191src/objects/mail/cleanup.c File Reference	1097
5.191.1 Function Documentation	1097
5.191.1.1 mail_destroy_header	1097
5.191.1.2 mail_message_cleanup	1097
5.192src/objects/mail/counters.c File Reference	1099
5.192.1 Function Documentation	1099
5.192.1.1 mail_count_received	1099
5.192.1.2 mail_header_end	1099
5.193src/providers/consumers/counters.c File Reference	1100
5.193.1 Function Documentation	1100
5.193.1.1 stamp_counter_check	1100
5.193.1.2 stamp_counter_increment	1100
5.194src/objects/mail/headers.c File Reference	1101
5.194.1 Function Documentation	1101
5.194.1.1 mail_add_forward_headers	1101
5.194.1.2 mail_add_inbound_headers	1102
5.194.1.3 mail_add_outbound_headers	1102
5.194.1.4 mail_add_required_headers	1103
5.194.1.5 mail_header_fetch_all	1103
5.194.1.6 mail_header_fetch_cleaned	1103
5.194.1.7 mail_header_pop	1104
5.194.1.8 mail_headers	1104
5.194.1.9 mail_mod_subject	1105
5.194.1.10mail_store_header	1105
5.195src/providers/prime/transposition/binary/headers.c File Reference	1106
5.195.1 Function Documentation	1106
5.195.1.1 prime_header_encrypted_message_write	1106
5.195.1.2 prime_header_encrypted_org_key_write	1106

5.195.1.3	prime_header_encrypted_user_key_write	1106
5.195.1.4	prime_header_length	1106
5.195.1.5	prime_header_org_key_write	1107
5.195.1.6	prime_header_org_signet_write	1107
5.195.1.7	prime_header_read	1107
5.195.1.8	prime_header_user_key_write	1107
5.195.1.9	prime_header_user_signet_write	1108
5.195.1.10	prime_header_user_signing_request_write	1108
5.195.1.11	prime_header_write	1108
5.196	src/objects/mail/load_message.c File Reference	1109
5.196.1	Function Documentation	1109
5.196.1.1	mail_load_header	1109
5.196.1.2	mail_load_message	1109
5.196.1.3	mail_load_message_top	1110
5.197	src/objects/mail/mail.h File Reference	1111
5.197.1	Define Documentation	1115
5.197.1.1	MAIL_MIME_RECURSION_LIMIT	1115
5.197.1.2	MAIL_SIGNATURES_RECURSION_LIMIT	1115
5.197.2	Function Documentation	1116
5.197.2.1	mail_add_forward_headers	1116
5.197.2.2	mail_add_inbound_headers	1116
5.197.2.3	mail_add_outbound_headers	1117
5.197.2.4	mail_add_required_headers	1117
5.197.2.5	mail_build_signature	1117
5.197.2.6	mail_cache_destroy	1118
5.197.2.7	mail_cache_get	1118
5.197.2.8	mail_cache_reset	1119
5.197.2.9	mail_cache_set	1119
5.197.2.10	mail_cache_start	1119
5.197.2.11	mail_cache_stop	1119
5.197.2.12	mail_cache_thread_stop	1120
5.197.2.13	mail_copy_message	1120
5.197.2.14	mail_count_received	1120
5.197.2.15	mail_create_directory	1121
5.197.2.16	mail_create_message	1121
5.197.2.17	mail_db_delete_message	1121
5.197.2.18	mail_db_hide_message	1122
5.197.2.19	mail_db_insert_duplicate_message	1122
5.197.2.20	mail_db_insert_message	1122

5.197.2.21mail_db_update_message_folder	1123
5.197.2.22mail_destroy	1123
5.197.2.23mail_destroy_header	1124
5.197.2.24mail_destroy_message	1124
5.197.2.25mail_discover_encoding	1124
5.197.2.26mail_discover_insertion_point	1125
5.197.2.27mail_discover_type	1125
5.197.2.28mail_domain_get	1125
5.197.2.29mail_extract_address	1126
5.197.2.30mail_extract_tag	1126
5.197.2.31mail_get_boundary	1127
5.197.2.32mail_get_chunk	1127
5.197.2.33mail_header_end	1127
5.197.2.34mail_header_fetch_all	1128
5.197.2.35mail_header_fetch_cleaned	1128
5.197.2.36mail_header_pop	1128
5.197.2.37mail_headers	1129
5.197.2.38mail_insert_chunk_base64	1129
5.197.2.39mail_insert_chunk_text	1130
5.197.2.40mail_load_header	1130
5.197.2.41mail_load_message	1131
5.197.2.42mail_load_message_top	1131
5.197.2.43mail_message	1132
5.197.2.44mail_message_cleanup	1132
5.197.2.45mail_message_path	1132
5.197.2.46mail_mime_boundary	1133
5.197.2.47mail_mime_child	1133
5.197.2.48mail_mime_content_encoding	1133
5.197.2.49mail_mime_content_id	1134
5.197.2.50mail_mime_count	1134
5.197.2.51mail_mime_encode_part	1134
5.197.2.52mail_mime_encoding	1135
5.197.2.53mail_mime_free	1135
5.197.2.54mail_mime_generate_boundary	1136
5.197.2.55mail_mime_get_media_type	1136
5.197.2.56mail_mime_get_smtp_envelope	1136
5.197.2.57mail_mime_header	1137
5.197.2.58mail_mime_part	1137
5.197.2.59mail_mime_split	1138

5.197.2.60	mail_mime_type	1138
5.197.2.61	mail_mime_type_group	1138
5.197.2.62	mail_mime_type_parameters	1139
5.197.2.63	mail_mime_type_parameters_key	1139
5.197.2.64	mail_mime_type_parameters_value	1139
5.197.2.65	mail_mime_type_sub	1140
5.197.2.66	mail_mime_update	1140
5.197.2.67	mail_mod_subject	1140
5.197.2.68	mail_modify_part	1141
5.197.2.69	mail_move_message	1141
5.197.2.70	mail_path_finder	1142
5.197.2.71	mail_remove_message	1142
5.197.2.72	mail_signature_add	1142
5.197.2.73	mail_store_header	1143
5.197.2.74	mail_store_message	1143
5.197.2.75	mail_store_message_data	1144
5.198	src/objects/mail/mime.c File Reference	1145
5.198.1	Function Documentation	1146
5.198.1.1	mail_mime_boundary	1146
5.198.1.2	mail_mime_child	1146
5.198.1.3	mail_mime_content_encoding	1147
5.198.1.4	mail_mime_content_id	1147
5.198.1.5	mail_mime_count	1147
5.198.1.6	mail_mime_encode_part	1148
5.198.1.7	mail_mime_encoding	1148
5.198.1.8	mail_mime_free	1149
5.198.1.9	mail_mime_generate_boundary	1149
5.198.1.10	mail_mime_get_media_type	1149
5.198.1.11	mail_mime_get_smtp_envelope	1150
5.198.1.12	mail_mime_header	1150
5.198.1.13	mail_mime_part	1150
5.198.1.14	mail_mime_split	1151
5.198.1.15	mail_mime_type	1151
5.198.1.16	mail_mime_type_group	1152
5.198.1.17	mail_mime_type_parameters	1152
5.198.1.18	mail_mime_type_parameters_key	1152
5.198.1.19	mail_mime_type_parameters_value	1153
5.198.1.20	mail_mime_type_sub	1153
5.198.1.21	mail_mime_update	1153

5.198.2 Variable Documentation	1154
5.198.2.1 media_types	1154
5.199src/objects/mail/objects.c File Reference	1155
5.199.1 Function Documentation	1155
5.199.1.1 mail_create_message	1155
5.199.1.2 mail_destroy	1155
5.199.1.3 mail_destroy_message	1156
5.199.1.4 mail_message	1156
5.200src/objects/objects.c File Reference	1157
5.200.1 Function Documentation	1157
5.200.1.1 obj_cache_prune	1157
5.200.1.2 obj_cache_start	1157
5.200.1.3 obj_cache_stop	1158
5.200.2 Variable Documentation	1158
5.200.2.1 objects	1158
5.201src/providers/prime/transposition/binary/objects.c File Reference	1159
5.201.1 Function Documentation	1159
5.201.1.1 prime_object_alloc	1159
5.201.1.2 prime_object_free	1159
5.201.1.3 prime_object_size_max	1159
5.201.1.4 prime_object_size_min	1159
5.201.1.5 prime_object_type	1160
5.201.2 Variable Documentation	1160
5.201.2.1 prime_types	1160
5.202src/objects/mail/parsing.c File Reference	1161
5.202.1 Function Documentation	1161
5.202.1.1 mail_domain_get	1161
5.202.1.2 mail_extract_address	1161
5.203src/objects/mail/paths.c File Reference	1162
5.203.1 Function Documentation	1162
5.203.1.1 mail_create_directory	1162
5.203.1.2 mail_message_path	1162
5.203.1.3 mail_path_finder	1163
5.204src/objects/mail/remove_message.c File Reference	1164
5.204.1 Function Documentation	1164
5.204.1.1 mail_remove_message	1164
5.205src/objects/mail/signatures.c File Reference	1165
5.205.1 Function Documentation	1165
5.205.1.1 mail_build_signature	1165

5.205.1.2 mail_discover_encoding	1166
5.205.1.3 mail_discover_insertion_point	1166
5.205.1.4 mail_discover_type	1167
5.205.1.5 mail_extract_tag	1167
5.205.1.6 mail_get_boundary	1168
5.205.1.7 mail_get_chunk	1168
5.205.1.8 mail_insert_chunk_base64	1168
5.205.1.9 mail_insert_chunk_text	1169
5.205.1.10mail_modify_part	1170
5.205.1.11mail_signature_add	1170
5.206src/objects/mail/store_message.c File Reference	1172
5.206.1 Function Documentation	1172
5.206.1.1 mail_copy_message	1172
5.206.1.2 mail_move_message	1173
5.206.1.3 mail_store_message	1173
5.206.1.4 mail_store_message_data	1173
5.207src/objects/messages/messages.h File Reference	1175
5.207.1 Define Documentation	1176
5.207.1.1 FMESSAGE_MAGIC_1	1176
5.207.1.2 FMESSAGE_MAGIC_2	1176
5.207.1.3 FMESSAGE_OPT_COMPRESSED	1177
5.207.1.4 FMESSAGE_OPT_ENCRYPTED	1177
5.207.1.5 MAIL_STATUS_ALL_FLAGS	1177
5.207.1.6 MAIL_STATUS_SYSTEM_FLAGS	1177
5.207.1.7 MAIL_STATUS_USER_FLAGS	1177
5.207.1.8 MESSAGE_ENCODING_7BIT	1177
5.207.1.9 MESSAGE_ENCODING_8BIT	1177
5.207.1.10MESSAGE_ENCODING_BASE64	1177
5.207.1.11MESSAGE_ENCODING_QUOTED_PRINTABLE	1178
5.207.1.12MESSAGE_ENCODING_UNKNOWN	1178
5.207.1.13MESSAGE_TYPE_HTML	1178
5.207.1.14MESSAGE_TYPE_MULTI_ALTERNATIVE	1178
5.207.1.15MESSAGE_TYPE_MULTI_MIXED	1178
5.207.1.16MESSAGE_TYPE_MULTI_RELATED	1178
5.207.1.17MESSAGE_TYPE_MULTI_RFC822	1178
5.207.1.18MESSAGE_TYPE_MULTI_UNKOWN	1178
5.207.1.19MESSAGE_TYPE_PLAIN	1178
5.207.1.20MESSAGE_TYPE_UNKNOWN	1179
5.207.2 Enumeration Type Documentation	1179

5.207.2.1 "@69	1179
5.207.3 Function Documentation	1179
5.207.3.1 __attribute__	1179
5.207.3.2 message_alloc	1179
5.207.3.3 message_free	1180
5.207.3.4 messages_update	1180
5.207.3.5 meta_data_fetch_folder_messages	1180
5.207.3.6 meta_data_fetch_message_tags	1181
5.207.3.7 meta_data_fetch_messages	1181
5.207.3.8 meta_message_by_number	1182
5.207.3.9 meta_message_dupe	1182
5.207.3.10 meta_message_free	1182
5.207.3.11 meta_messages_copier	1182
5.207.3.12 meta_messages_login_update	1183
5.207.3.13 meta_messages_mover	1183
5.207.3.14 meta_messages_update	1184
5.207.3.15 meta_messages_update_sequences	1185
5.207.4 Variable Documentation	1185
5.207.4.1 message_header_t	1185
5.207.4.2 message_t	1185
5.208src/providers/prime/messages/messages.h File Reference	1186
5.208.1 Function Documentation	1186
5.208.1.1 encrypted_message_alloc	1186
5.208.1.2 encrypted_message_cleanup	1186
5.208.1.3 encrypted_message_free	1186
5.208.1.4 naked_message_get	1186
5.208.1.5 naked_message_set	1187
5.209src/objects/messages/meta.c File Reference	1188
5.209.1 Function Documentation	1188
5.209.1.1 meta_message_by_number	1188
5.209.1.2 meta_message_dupe	1188
5.209.1.3 meta_message_free	1189
5.209.1.4 meta_messages_copier	1189
5.209.1.5 meta_messages_login_update	1190
5.209.1.6 meta_messages_mover	1190
5.209.1.7 meta_messages_update	1191
5.209.1.8 meta_messages_update_sequences	1191
5.210src/objects/meta/meta.c File Reference	1192
5.210.1 Function Documentation	1192

5.210.1.1 meta_alloc	1192
5.210.1.2 meta_free	1192
5.210.1.3 meta_get	1192
5.211src/objects/meta/alerts.c File Reference	1194
5.211.1 Function Documentation	1194
5.211.1.1 alert_alloc	1194
5.212src/objects/meta/alias.c File Reference	1195
5.212.1 Function Documentation	1195
5.212.1.1 alias_alloc	1195
5.213src/objects/meta/crypto.c File Reference	1196
5.213.1 Function Documentation	1196
5.213.1.1 meta_crypto_keys_create	1196
5.214src/providers/dime/common/crypto.c File Reference	1197
5.214.1 Function Documentation	1199
5.214.1.1 _compute_aes256_kek	1199
5.214.1.2 _crypto_init	1199
5.214.1.3 _crypto_shutdown	1199
5.214.1.4 _decrypt_aes_256	1199
5.214.1.5 _deserialize_ec_privkey	1200
5.214.1.6 _deserialize_ec_pubkey	1200
5.214.1.7 _deserialize_ed25519_privkey	1200
5.214.1.8 _deserialize_ed25519_pubkey	1200
5.214.1.9 _ec_sign_data	1201
5.214.1.10 _ec_sign_sha_data	1201
5.214.1.11 _ecies_env_derivation	1202
5.214.1.12 _ed25519_sign_data	1202
5.214.1.13 _ed25519_verify_sig	1202
5.214.1.14 _encrypt_aes_256	1202
5.214.1.15 _free_ec_key	1203
5.214.1.16 _free_ed25519_key	1203
5.214.1.17 _free_ed25519_key_chain	1203
5.214.1.18 _generate_ec_keypair	1203
5.214.1.19 _generate_ed25519_keypair	1203
5.214.1.20 _get_random_bytes	1204
5.214.1.21 _load_ec_privkey	1204
5.214.1.22 _load_ec_pubkey	1204
5.214.1.23 _load_ed25519_privkey	1204
5.214.1.24 _serialize_ec_privkey	1205
5.214.1.25 _serialize_ec_pubkey	1205

5.214.1.26_verify_ec_sha_signature	1205
5.214.1.27_verify_ec_signature	1206
5.215src/providers/stacie/crypto.c File Reference	1207
5.215.1 Detailed Description	1207
5.215.2 Function Documentation	1207
5.215.2.1 stacie_decrypt	1207
5.215.2.2 stacie_encrypt	1207
5.216src/objects/meta/indexes.c File Reference	1209
5.216.1 Function Documentation	1209
5.216.1.1 meta_inx_find	1209
5.216.1.2 meta_inx_remove	1209
5.217src/objects/meta/locking.c File Reference	1210
5.217.1 Function Documentation	1210
5.217.1.1 meta_user_rlock	1210
5.217.1.2 meta_user_unlock	1210
5.217.1.3 meta_user_wlock	1211
5.218src/objects/meta/references.c File Reference	1212
5.218.1 Function Documentation	1212
5.218.1.1 meta_user_ref_add	1212
5.218.1.2 meta_user_ref_dec	1212
5.218.1.3 meta_user_ref_protocol_total	1213
5.218.1.4 meta_user_ref_stamp	1213
5.218.1.5 meta_user_ref_total	1213
5.219src/objects/meta/serials.c File Reference	1215
5.219.1 Function Documentation	1215
5.219.1.1 meta_user_serial_check	1215
5.219.1.2 meta_user_serial_get	1215
5.219.1.3 meta_user_serial_set	1216
5.220src/objects/serials.c File Reference	1217
5.220.1 Function Documentation	1217
5.220.1.1 serial_get	1217
5.220.1.2 serial_increment	1217
5.220.1.3 serial_prefix	1218
5.220.1.4 serial_reset	1218
5.220.2 Variable Documentation	1219
5.220.2.1 serial_prefix_strings	1219
5.221src/objects/meta/updaters.c File Reference	1220
5.221.1 Function Documentation	1220
5.221.1.1 meta_update_aliases	1220

5.221.1.2 meta_update_contacts	1221
5.221.1.3 meta_update_folders	1221
5.221.1.4 meta_update_keys	1221
5.221.1.5 meta_update_message_folders	1222
5.221.1.6 meta_update_realms	1222
5.221.1.7 meta_update_user	1223
5.222src/objects/objects.h File Reference	1224
5.222.1 Enumeration Type Documentation	1225
5.222.1.1 "@70	1225
5.222.2 Function Documentation	1225
5.222.2.1 lock_get	1225
5.222.2.2 lock_release	1226
5.222.2.3 obj_cache_prune	1226
5.222.2.4 obj_cache_start	1226
5.222.2.5 obj_cache_stop	1227
5.222.2.6 serial_get	1227
5.222.2.7 serial_increment	1227
5.222.2.8 serial_reset	1228
5.222.2.9 user_lock	1228
5.222.2.10user_unlock	1228
5.222.3 Variable Documentation	1228
5.222.3.1 objects	1228
5.223src/objects/sessions/sessions.c File Reference	1229
5.223.1 Function Documentation	1230
5.223.1.1 sess_create	1230
5.223.1.2 sess_destroy	1230
5.223.1.3 sess_get	1231
5.223.1.4 sess_key	1231
5.223.1.5 sess_number	1231
5.223.1.6 sess_ref_add	1232
5.223.1.7 sess_ref_dec	1232
5.223.1.8 sess_ref_stamp	1232
5.223.1.9 sess_ref_total	1233
5.223.1.10sess_refresh_check	1233
5.223.1.11sess_refresh_flush	1233
5.223.1.12sess_refresh_stamp	1234
5.223.1.13sess_release	1234
5.223.1.14sess_release_attachment	1234
5.223.1.15sess_release_composition	1235

5.223.1.16	sess_serial_check	1235
5.223.1.17	sess_token	1235
5.223.1.18	sess_trigger	1236
5.223.1.19	sess_update	1236
5.223.2	Variable Documentation	1236
5.223.2.1	lock	1236
5.223.2.2	number	1236
5.223.2.3	sessions	1237
5.224	src/servers/http/sessions.c File Reference	1238
5.224.1	Function Documentation	1238
5.224.1.1	http_session_destroy	1238
5.224.1.2	http_session_reset	1238
5.225	src/servers/imap/sessions.c File Reference	1239
5.225.1	Function Documentation	1239
5.225.1.1	imap_session_destroy	1239
5.225.1.2	imap_session_update	1239
5.226	src/servers/molten/sessions.c File Reference	1240
5.226.1	Function Documentation	1240
5.226.1.1	molten_session_destroy	1240
5.227	src/servers/pop/sessions.c File Reference	1241
5.227.1	Function Documentation	1241
5.227.1.1	pop_session_destroy	1241
5.227.1.2	pop_session_reset	1241
5.228	src/web/register/sessions.c File Reference	1242
5.228.1	Function Documentation	1242
5.228.1.1	register_session_cache	1242
5.228.1.2	register_session_free	1242
5.228.1.3	register_session_generate	1243
5.228.1.4	register_session_get	1243
5.229	src/objects/warehouse/domains.c File Reference	1244
5.229.1	Function Documentation	1244
5.229.1.1	domain_alloc	1244
5.229.1.2	domain_dkim	1245
5.229.1.3	domain_mailboxes	1245
5.229.1.4	domain_restricted	1245
5.229.1.5	domain_spf	1246
5.229.1.6	domain_start	1246
5.229.1.7	domain_stop	1246
5.229.1.8	domain_wildcard	1246

5.229.2 Variable Documentation	1247
5.229.2.1 domains	1247
5.230src/objects/warehouse/patterns.c File Reference	1248
5.230.1 Function Documentation	1248
5.230.1.1 pattern_check	1248
5.230.1.2 pattern_start	1248
5.230.1.3 pattern_stop	1249
5.230.1.4 pattern_update	1249
5.230.2 Variable Documentation	1249
5.230.2.1 patterns_list	1249
5.230.2.2 patterns_mutex	1249
5.230.2.3 patterns_stamp	1249
5.231src/objects/warehouse/warehouse.c File Reference	1250
5.231.1 Function Documentation	1250
5.231.1.1 warehouse_start	1250
5.231.1.2 warehouse_stop	1250
5.231.1.3 warehouse_update	1251
5.232src/objects/warehouse/warehouse.h File Reference	1252
5.232.1 Function Documentation	1253
5.232.1.1 __attribute__	1253
5.232.1.2 domain_alloc	1253
5.232.1.3 domain_dkim	1253
5.232.1.4 domain_mailboxes	1254
5.232.1.5 domain_restricted	1254
5.232.1.6 domain_spf	1254
5.232.1.7 domain_start	1254
5.232.1.8 domain_stop	1255
5.232.1.9 domain_wildcard	1255
5.232.1.10pattern_check	1255
5.232.1.11pattern_start	1255
5.232.1.12pattern_stop	1256
5.232.1.13pattern_update	1256
5.232.1.14warehouse_fetch_domains	1256
5.232.1.15warehouse_fetch_patterns	1256
5.232.1.16warehouse_start	1257
5.232.1.17warehouse_stop	1257
5.232.1.18warehouse_update	1257
5.232.2 Variable Documentation	1258
5.232.2.1 domain_t	1258

5.233src/providers/checkers/allocations.h File Reference	1259
5.233.1 Variable Documentation	1259
5.233.1.1 comment	1259
5.233.1.2 ip_v4_allocations_t	1259
5.233.1.3 owner	1259
5.233.1.4 status	1259
5.233.1.5 weight	1259
5.234src/providers/checkers/checkers.h File Reference	1260
5.234.1 Define Documentation	1261
5.234.1.1 IP_RANDOMIZER_POOL	1261
5.234.1.2 IP_RANDOMIZER_PUSH_MAX	1261
5.234.1.3 IP_RANDOMIZER_PUSH_MIN	1261
5.234.2 Enumeration Type Documentation	1262
5.234.2.1 "@73	1262
5.234.2.2 "@74	1262
5.234.2.3 "@75	1262
5.234.3 Function Documentation	1262
5.234.3.1 dkim_memory_alloc	1262
5.234.3.2 dkim_memory_free	1263
5.234.3.3 dkim_signature_create	1263
5.234.3.4 dkim_signature_verify	1263
5.234.3.5 dkim_start	1264
5.234.3.6 dkim_stop	1264
5.234.3.7 dspam_check	1264
5.234.3.8 dspam_start	1265
5.234.3.9 dspam_stop	1265
5.234.3.10dspam_train	1265
5.234.3.11lib_load_clamav	1265
5.234.3.12lib_load_dkim	1265
5.234.3.13lib_load_dspam	1266
5.234.3.14lib_load_spf	1266
5.234.3.15lib_version_clamav	1266
5.234.3.16lib_version_dkim	1266
5.234.3.17lib_version_dspam	1267
5.234.3.18lib_version_spf	1267
5.234.3.19spf_check	1267
5.234.3.20spf_start	1268
5.234.3.21spf_stop	1268
5.234.3.22virus_check	1268

5.234.3.23virus_engine_create	1268
5.234.3.24virus_engine_destroy	1269
5.234.3.25virus_engine_refresh	1269
5.234.3.26virus_sigs_loaded	1269
5.234.3.27virus_sigs_total	1269
5.234.3.28virus_start	1270
5.234.3.29virus_stop	1270
5.235src/providers/checkers/clamav.c File Reference	1271
5.235.1 Function Documentation	1271
5.235.1.1 lib_load_clamav	1271
5.235.1.2 lib_version_clamav	1272
5.235.1.3 virus_check	1272
5.235.1.4 virus_engine_create	1272
5.235.1.5 virus_engine_destroy	1272
5.235.1.6 virus_engine_refresh	1273
5.235.1.7 virus_sigs_loaded	1273
5.235.1.8 virus_sigs_total	1273
5.235.1.9 virus_start	1273
5.235.1.10virus_stop	1274
5.235.2 Variable Documentation	1274
5.235.2.1 virus_engine	1274
5.235.2.2 virus_lock	1274
5.235.2.3 virus_sigs	1274
5.235.2.4 virus_spool	1274
5.235.2.5 virus_stat	1275
5.236src/providers/checkers/dkim.c File Reference	1276
5.236.1 Define Documentation	1276
5.236.1.1 DKIM_PROCESS_ALL	1276
5.236.2 Function Documentation	1277
5.236.2.1 dkim_memory_alloc	1277
5.236.2.2 dkim_memory_free	1277
5.236.2.3 dkim_signature_create	1277
5.236.2.4 dkim_signature_verify	1278
5.236.2.5 dkim_start	1278
5.236.2.6 dkim_stop	1279
5.236.2.7 lib_load_dkim	1279
5.236.2.8 lib_version_dkim	1279
5.236.3 Variable Documentation	1279
5.236.3.1 dkim_engine	1279

5.236.3.2 dkim_version	1279
5.237src/providers/checkers/dspam.c File Reference	1280
5.237.1 Function Documentation	1280
5.237.1.1 dspam_check	1280
5.237.1.2 dspam_start	1280
5.237.1.3 dspam_stop	1280
5.237.1.4 dspam_train	1281
5.237.1.5 lib_load_dspam	1281
5.237.1.6 lib_version_dspam	1281
5.237.2 Variable Documentation	1281
5.237.2.1 sql_pool	1281
5.238src/providers/checkers/spf.c File Reference	1283
5.238.1 Function Documentation	1283
5.238.1.1 lib_load_spf	1283
5.238.1.2 lib_version_spf	1283
5.238.1.3 spf_check	1284
5.238.1.4 spf_start	1284
5.238.1.5 spf_stop	1284
5.238.2 Variable Documentation	1284
5.238.2.1 spf_pool	1284
5.238.2.2 spf_version	1285
5.239src/providers/compress/bzip.c File Reference	1286
5.239.1 Function Documentation	1286
5.239.1.1 compress_bzip	1286
5.239.1.2 decompress_bzip	1286
5.239.1.3 lib_load_bzip	1287
5.239.1.4 lib_version_bzip	1287
5.239.2 Variable Documentation	1287
5.239.2.1 bzip_version	1287
5.240src/providers/compress/compress.c File Reference	1288
5.240.1 Function Documentation	1288
5.240.1.1 compress_alloc	1288
5.240.1.2 compress_block_length	1289
5.240.1.3 compress_body_data	1289
5.240.1.4 compress_body_hash	1289
5.240.1.5 compress_body_length	1290
5.240.1.6 compress_body_offset	1290
5.240.1.7 compress_cleanup	1290
5.240.1.8 compress_free	1290

5.240.1.9 compress_import	1291
5.240.1.10compress_orig_hash	1291
5.240.1.11compress_orig_length	1291
5.240.1.12compress_total_length	1291
5.241src/providers/compress/compress.h File Reference	1292
5.241.1 Typedef Documentation	1293
5.241.1.1 compress_t	1293
5.241.2 Enumeration Type Documentation	1294
5.241.2.1 "@76	1294
5.241.3 Function Documentation	1294
5.241.3.1 __attribute__	1294
5.241.3.2 compress_alloc	1294
5.241.3.3 compress_block_length	1294
5.241.3.4 compress_body_data	1294
5.241.3.5 compress_body_hash	1295
5.241.3.6 compress_body_length	1295
5.241.3.7 compress_body_offset	1295
5.241.3.8 compress_bzip	1295
5.241.3.9 compress_cleanup	1296
5.241.3.10compress_free	1296
5.241.3.11compress_import	1296
5.241.3.12compress_lzo	1297
5.241.3.13compress_orig_hash	1297
5.241.3.14compress_orig_length	1297
5.241.3.15compress_total_length	1298
5.241.3.16compress_zlib	1298
5.241.3.17decompress_block_lzo	1298
5.241.3.18decompress_bzip	1298
5.241.3.19decompress_lzo	1299
5.241.3.20decompress_zlib	1299
5.241.3.21engine_compress	1299
5.241.3.22engine_decompress	1299
5.241.3.23lib_load_bzip	1300
5.241.3.24lib_load_lzo	1300
5.241.3.25lib_load_zlib	1300
5.241.3.26lib_version_bzip	1300
5.241.3.27lib_version_lzo	1300
5.241.3.28lib_version_zlib	1301
5.241.4 Variable Documentation	1301

5.241.4.1 COMPRESS_ENGINE	1301
5.241.4.2 compress_head_t	1301
5.242src/providers/compress/engine.c File Reference	1302
5.242.1 Function Documentation	1302
5.242.1.1 engine_compress	1302
5.242.1.2 engine_decompress	1302
5.243src/providers/compress/lzo.c File Reference	1303
5.243.1 Function Documentation	1303
5.243.1.1 compress_lzo	1303
5.243.1.2 decompress_block_lzo	1303
5.243.1.3 decompress_lzo	1304
5.243.1.4 lib_load_lzo	1304
5.243.1.5 lib_version_lzo	1304
5.244src/providers/compress/zlib.c File Reference	1305
5.244.1 Function Documentation	1305
5.244.1.1 compress_zlib	1305
5.244.1.2 decompress_zlib	1305
5.244.1.3 lib_load_zlib	1306
5.244.1.4 lib_version_zlib	1306
5.244.2 Variable Documentation	1306
5.244.2.1 deflate_d	1306
5.244.2.2 deflateEnd_d	1306
5.244.2.3 deflateInit2__d	1306
5.245src/providers/consumers/consumers.h File Reference	1307
5.245.1 Function Documentation	1309
5.245.1.1 __attribute__	1309
5.245.1.2 cache_add	1309
5.245.1.3 cache_append	1309
5.245.1.4 cache_decrement	1309
5.245.1.5 cache_delete	1310
5.245.1.6 cache_flush	1310
5.245.1.7 cache_get	1310
5.245.1.8 cache_get_u64	1311
5.245.1.9 cache_increment	1311
5.245.1.10cache_set	1311
5.245.1.11lcache_set_u64	1312
5.245.1.12cache_silent_add	1312
5.245.1.13cache_start	1312
5.245.1.14cache_stop	1313

5.245.1.15	deserialize_count_digits	1313
5.245.1.16	deserialize_int16	1313
5.245.1.17	deserialize_int32	1313
5.245.1.18	deserialize_int64	1313
5.245.1.19	deserialize_ns	1314
5.245.1.20	deserialize_ssz	1314
5.245.1.21	deserialize_st	1314
5.245.1.22	deserialize_sz	1315
5.245.1.23	deserialize_uint16	1315
5.245.1.24	deserialize_uint32	1315
5.245.1.25	deserialize_uint64	1315
5.245.1.26	lib_load_cache	1316
5.245.1.27	lib_version_cache	1316
5.245.1.28	serialize_int16	1316
5.245.1.29	serialize_int32	1317
5.245.1.30	serialize_int64	1317
5.245.1.31	serialize_ns	1317
5.245.1.32	serialize_ssz	1318
5.245.1.33	serialize_st	1318
5.245.1.34	serialize_sz	1319
5.245.1.35	serialize_uint16	1319
5.245.1.36	serialize_uint32	1319
5.245.1.37	serialize_uint64	1320
5.245.2	Variable Documentation	1320
5.245.2.1	serialization_t	1320
5.246	src/providers/consumers/deserialization.c File Reference	1321
5.246.1	Function Documentation	1321
5.246.1.1	deserialize_count_digits	1321
5.246.1.2	deserialize_int16	1322
5.246.1.3	deserialize_int32	1322
5.246.1.4	deserialize_int64	1322
5.246.1.5	deserialize_ns	1323
5.246.1.6	deserialize_ssz	1323
5.246.1.7	deserialize_st	1323
5.246.1.8	deserialize_sz	1324
5.246.1.9	deserialize_uint16	1324
5.246.1.10	deserialize_uint32	1324
5.246.1.11	deserialize_uint64	1325
5.247	src/providers/consumers/serialization.c File Reference	1326

5.247.1 Function Documentation	1326
5.247.1.1 serialize_int16	1326
5.247.1.2 serialize_int32	1327
5.247.1.3 serialize_int64	1327
5.247.1.4 serialize_ns	1327
5.247.1.5 serialize_ssz	1328
5.247.1.6 serialize_st	1328
5.247.1.7 serialize_sz	1329
5.247.1.8 serialize_uint16	1329
5.247.1.9 serialize_uint32	1329
5.247.1.10serialize_uint64	1330
5.248src/providers/cryptography/ciphers.c File Reference	1331
5.248.1 Function Documentation	1331
5.248.1.1 cipher_block_length	1331
5.248.1.2 cipher_id	1331
5.248.1.3 cipher_key_length	1332
5.248.1.4 cipher_name	1332
5.248.1.5 cipher_numeric_id	1332
5.248.1.6 cipher_vector_length	1333
5.249src/providers/cryptography/cryptex.c File Reference	1334
5.249.1 Function Documentation	1334
5.249.1.1 deprecated_cryptex_alloc	1334
5.249.1.2 deprecated_cryptex_body_data	1334
5.249.1.3 deprecated_cryptex_body_length	1335
5.249.1.4 deprecated_cryptex_envelope_data	1335
5.249.1.5 deprecated_cryptex_envelope_length	1335
5.249.1.6 deprecated_cryptex_free	1336
5.249.1.7 deprecated_cryptex_hmac_data	1336
5.249.1.8 deprecated_cryptex_hmac_length	1336
5.249.1.9 deprecated_cryptex_original_length	1337
5.249.1.10deprecated_cryptex_total_length	1337
5.250src/providers/deprecated/cryptex.c File Reference	1338
5.250.1 Function Documentation	1338
5.250.1.1 cryptex_alloc	1338
5.250.1.2 cryptex_body_data	1339
5.250.1.3 cryptex_body_length	1339
5.250.1.4 cryptex_envelope_data	1339
5.250.1.5 cryptex_envelope_length	1339
5.250.1.6 cryptex_free	1340

5.250.1.7 cryptex_hmac_data	1340
5.250.1.8 cryptex_hmac_length	1340
5.250.1.9 cryptex_original_length	1341
5.250.1.10 cryptex_total_length	1341
5.251 src/providers/cryptography/cryptography.h File Reference	1342
5.251.1 Define Documentation	1348
5.251.1.1 BN_num_bytes_d	1348
5.251.1.2 ECIES_CIPHER	1348
5.251.1.3 ECIES_CURVE	1348
5.251.1.4 ECIES_ENVELOPE	1348
5.251.1.5 ECIES_HMAC	1348
5.251.1.6 MAGMA_CIPHERS_GENERIC	1348
5.251.1.7 MAGMA_CIPHERS_HIGH	1348
5.251.1.8 MAGMA_CIPHERS_LOW	1349
5.251.1.9 MAGMA_CIPHERS_MEDIUM	1349
5.251.1.10 OPENSSL_free_d	1349
5.251.2 Typedef Documentation	1349
5.251.2.1 cipher_t	1349
5.251.2.2 cryptex_t	1349
5.251.2.3 digest_t	1349
5.251.2.4 scramble_t	1349
5.251.2.5 TLS	1349
5.251.3 Enumeration Type Documentation	1349
5.251.3.1 ECIES_KEY_TYPE	1349
5.251.4 Function Documentation	1350
5.251.4.1 __attribute__	1350
5.251.4.2 cipher_block_length	1350
5.251.4.3 cipher_id	1350
5.251.4.4 cipher_key_length	1350
5.251.4.5 cipher_name	1351
5.251.4.6 cipher_numeric_id	1351
5.251.4.7 cipher_vector_length	1351
5.251.4.8 deprecated_cryptex_alloc	1352
5.251.4.9 deprecated_cryptex_body_data	1352
5.251.4.10 deprecated_cryptex_body_length	1352
5.251.4.11 deprecated_cryptex_envelope_data	1352
5.251.4.12 deprecated_cryptex_envelope_length	1353
5.251.4.13 deprecated_cryptex_free	1353
5.251.4.14 deprecated_cryptex_hmac_data	1353

5.251.4.15	deprecated_cryptex_hmac_length	1353
5.251.4.16	deprecated_cryptex_original_length	1354
5.251.4.17	deprecated_cryptex_total_length	1354
5.251.4.18	deprecated_ecies_decrypt	1354
5.251.4.19	deprecated_ecies_encrypt	1355
5.251.4.20	deprecated_ecies_envelope_derivation	1355
5.251.4.21	deprecated_ecies_group	1355
5.251.4.22	deprecated_ecies_key_alloc	1355
5.251.4.23	deprecated_ecies_key_create	1356
5.251.4.24	deprecated_ecies_key_free	1356
5.251.4.25	deprecated_ecies_key_private	1356
5.251.4.26	deprecated_ecies_key_private_bin	1356
5.251.4.27	deprecated_ecies_key_private_hex	1357
5.251.4.28	deprecated_ecies_key_public	1357
5.251.4.29	deprecated_ecies_key_public_bin	1357
5.251.4.30	deprecated_ecies_key_public_hex	1357
5.251.4.31	deprecated_ecies_start	1358
5.251.4.32	deprecated_ecies_stop	1358
5.251.4.33	deprecated_hmac_digest	1358
5.251.4.34	deprecated_hmac_md4	1359
5.251.4.35	deprecated_hmac_md5	1359
5.251.4.36	deprecated_hmac_ripemd160	1359
5.251.4.37	deprecated_hmac_sha	1359
5.251.4.38	deprecated_hmac_sha1	1359
5.251.4.39	deprecated_hmac_sha224	1359
5.251.4.40	deprecated_hmac_sha256	1359
5.251.4.41	deprecated_hmac_sha384	1359
5.251.4.42	deprecated_hmac_sha512	1359
5.251.4.43	deprecated_scramble_alloc	1360
5.251.4.44	deprecated_scramble_body_data	1360
5.251.4.45	deprecated_scramble_body_hash	1360
5.251.4.46	deprecated_scramble_body_length	1360
5.251.4.47	deprecated_scramble_cleanup	1361
5.251.4.48	deprecated_scramble_decrypt	1361
5.251.4.49	deprecated_scramble_encrypt	1361
5.251.4.50	deprecated_scramble_free	1362
5.251.4.51	deprecated_scramble_import	1362
5.251.4.52	deprecated_scramble_orig_length	1362
5.251.4.53	deprecated_scramble_total_length	1363

5.251.4.54	deprecated_scramble_vector_data	1363
5.251.4.55	deprecated_scramble_vector_length	1363
5.251.4.56	deprecated_symmetric_decrypt	1364
5.251.4.57	deprecated_symmetric_encrypt	1364
5.251.4.58	deprecated_symmetric_key	1364
5.251.4.59	deprecated_symmetric_vector	1365
5.251.4.60	dh_exchange_2048	1365
5.251.4.61	dh_exchange_4096	1365
5.251.4.62	dh_params_2048	1366
5.251.4.63	dh_params_4096	1366
5.251.4.64	dh_params_generate	1366
5.251.4.65	dh_params_generate_callback	1366
5.251.4.66	dh_static_2048	1366
5.251.4.67	dh_static_4096	1366
5.251.4.68	digest_id	1367
5.251.4.69	digest_name	1367
5.251.4.70	hash_digest	1367
5.251.4.71	hash_md4	1367
5.251.4.72	hash_md5	1367
5.251.4.73	hash_ripemd160	1367
5.251.4.74	hash_sha	1367
5.251.4.75	hash_sha1	1367
5.251.4.76	hash_sha224	1368
5.251.4.77	hash_sha256	1368
5.251.4.78	hash_sha384	1368
5.251.4.79	hash_sha512	1368
5.251.4.80	lib_load_openssl	1368
5.251.4.81	lib_version_openssl	1368
5.251.4.82	rand_choices	1369
5.251.4.83	rand_get_int16	1369
5.251.4.84	rand_get_int32	1369
5.251.4.85	rand_get_int64	1369
5.251.4.86	rand_get_int8	1369
5.251.4.87	rand_get_uint16	1370
5.251.4.88	rand_get_uint32	1370
5.251.4.89	rand_get_uint64	1370
5.251.4.90	rand_get_uint8	1371
5.251.4.91	rand_start	1371
5.251.4.92	rand_stop	1371

5.251.4.93	rand_thread_start	1371
5.251.4.94	rand_write	1371
5.251.4.95	ssl_dh2048_exchange_callback	1372
5.251.4.96	ssl_dh4096_exchange_callback	1372
5.251.4.97	ssl_dh_generate_callback	1372
5.251.4.98	ssl_ecdh_exchange_callback	1372
5.251.4.99	ssl_error_string	1372
5.251.4.100	ssl_locking_callback	1373
5.251.4.101	ssl_start	1373
5.251.4.102	ssl_stop	1374
5.251.4.103	ssl_thread_id_callback	1374
5.251.4.104	ssl_thread_stop	1374
5.251.4.105	ssl_verify_privkey	1374
5.251.4.106	ssl_bits	1375
5.251.4.107	ssl_cipher	1375
5.251.4.108	ssl_client_alloc	1375
5.251.4.109	ssl_continue	1376
5.251.4.110	ssl_error	1376
5.251.4.111	ssl_free	1376
5.251.4.112	ssl_print	1376
5.251.4.113	ssl_read	1377
5.251.4.114	ssl_server_alloc	1377
5.251.4.115	ssl_server_create	1377
5.251.4.116	ssl_server_destroy	1378
5.251.4.117	ssl_status	1378
5.251.4.118	ssl_suite	1378
5.251.4.119	ssl_version	1379
5.251.4.120	ssl_write	1379
5.251.5	Variable Documentation	1379
5.251.5.1	cryptex_head_t	1379
5.251.5.2	scramble_head_t	1380
5.252	src/providers/prime/cryptography/cryptography.h File Reference	1381
5.252.1	Function Documentation	1382
5.252.1.1	aes_artifact_decrypt	1382
5.252.1.2	aes_artifact_encrypt	1382
5.252.1.3	aes_chunk_decrypt	1382
5.252.1.4	aes_chunk_encrypt	1382
5.252.1.5	aes_cipher_key	1383
5.252.1.6	aes_tag_shard	1383

5.252.1.7 aes_vector_shard	1383
5.252.1.8 ed25519_alloc	1383
5.252.1.9 ed25519_free	1384
5.252.1.10ed25519_generate	1384
5.252.1.11ed25519_private_get	1384
5.252.1.12ed25519_private_set	1384
5.252.1.13ed25519_public_get	1384
5.252.1.14ed25519_public_set	1384
5.252.1.15ed25519_sign	1385
5.252.1.16ed25519_type	1385
5.252.1.17ed25519_verify	1385
5.252.1.18secp256k1_alloc	1385
5.252.1.19secp256k1_compute_kek	1386
5.252.1.20secp256k1_free	1386
5.252.1.21secp256k1_generate	1386
5.252.1.22secp256k1_private_get	1386
5.252.1.23secp256k1_private_set	1387
5.252.1.24secp256k1_public_get	1387
5.252.1.25secp256k1_public_set	1387
5.252.1.26secp256k1_type	1388
5.253src/providers/cryptography/digest.c File Reference	1389
5.253.1 Function Documentation	1389
5.253.1.1 digest_id	1389
5.253.1.2 digest_length_output	1389
5.253.1.3 digest_name	1389
5.254src/providers/cryptography/ecies.c File Reference	1390
5.254.1 Function Documentation	1391
5.254.1.1 deprecated_ecies_decrypt	1391
5.254.1.2 deprecated_ecies_encrypt	1391
5.254.1.3 deprecated_ecies_envelope_derivation	1392
5.254.1.4 deprecated_ecies_group	1392
5.254.1.5 deprecated_ecies_key_alloc	1392
5.254.1.6 deprecated_ecies_key_create	1392
5.254.1.7 deprecated_ecies_key_free	1392
5.254.1.8 deprecated_ecies_key_private	1393
5.254.1.9 deprecated_ecies_key_private_bin	1393
5.254.1.10deprecated_ecies_key_private_hex	1393
5.254.1.11deprecated_ecies_key_public	1393
5.254.1.12deprecated_ecies_key_public_bin	1393

5.254.1.13	deprecated_ecies_key_public_hex	1394
5.254.1.14	deprecated_ecies_start	1394
5.254.1.15	deprecated_ecies_stop	1394
5.254.2	Variable Documentation	1394
5.254.2.1	ecies_cipher_evp	1394
5.254.2.2	ecies_curve_group	1395
5.254.2.3	ecies_envelope_evp	1395
5.254.2.4	ecies_hmac_evp	1395
5.255	src/providers/deprecated/ecies.c File Reference	1396
5.255.1	Function Documentation	1397
5.255.1.1	ecies_decrypt	1397
5.255.1.2	ecies_encrypt	1397
5.255.1.3	ecies_envelope_derivation	1397
5.255.1.4	ecies_group	1398
5.255.1.5	ecies_key_alloc	1398
5.255.1.6	ecies_key_create	1398
5.255.1.7	ecies_key_free	1398
5.255.1.8	ecies_key_private	1398
5.255.1.9	ecies_key_private_bin	1399
5.255.1.10	ecies_key_private_hex	1399
5.255.1.11	ecies_key_public	1399
5.255.1.12	ecies_key_public_bin	1399
5.255.1.13	ecies_key_public_hex	1400
5.255.1.14	ecies_start	1400
5.255.1.15	ecies_stop	1400
5.255.2	Variable Documentation	1400
5.255.2.1	ecies_cipher_evp	1400
5.255.2.2	ecies_curve_group	1400
5.255.2.3	ecies_envelope_evp	1400
5.255.2.4	ecies_hmac_evp	1401
5.256	src/providers/cryptography/hash.c File Reference	1402
5.256.1	Function Documentation	1402
5.256.1.1	hash_digest	1402
5.256.1.2	hash_md4	1402
5.256.1.3	hash_md5	1402
5.256.1.4	hash_ripemd160	1402
5.256.1.5	hash_sha	1403
5.256.1.6	hash_sha1	1403
5.256.1.7	hash_sha224	1403

5.256.1.8 hash_sha256	1403
5.256.1.9 hash_sha384	1403
5.256.1.10 hash_sha512	1403
5.257src/providers/cryptography/hmac.c File Reference	1404
5.257.1 Function Documentation	1404
5.257.1.1 deprecated_hmac_digest	1404
5.257.1.2 deprecated_hmac_md4	1404
5.257.1.3 deprecated_hmac_md5	1404
5.257.1.4 deprecated_hmac_ripemd160	1405
5.257.1.5 deprecated_hmac_sha	1405
5.257.1.6 deprecated_hmac_sha1	1405
5.257.1.7 deprecated_hmac_sha224	1405
5.257.1.8 deprecated_hmac_sha256	1405
5.257.1.9 deprecated_hmac_sha384	1405
5.257.1.10 deprecated_hmac_sha512	1405
5.258src/providers/deprecated/hmac.c File Reference	1406
5.258.1 Function Documentation	1406
5.258.1.1 hmac_digest	1406
5.258.1.2 hmac_md4	1406
5.258.1.3 hmac_md5	1407
5.258.1.4 hmac_ripemd160	1407
5.258.1.5 hmac_sha	1407
5.258.1.6 hmac_sha1	1407
5.258.1.7 hmac_sha224	1407
5.258.1.8 hmac_sha256	1407
5.258.1.9 hmac_sha384	1407
5.258.1.10 hmac_sha512	1407
5.259src/providers/cryptography/openssl.c File Reference	1408
5.259.1 Function Documentation	1409
5.259.1.1 lib_load_openssl	1409
5.259.1.2 lib_version_openssl	1409
5.259.1.3 ssl_ecdh_exchange_callback	1409
5.259.1.4 ssl_error_string	1409
5.259.1.5 ssl_locking_callback	1410
5.259.1.6 ssl_start	1410
5.259.1.7 ssl_stop	1411
5.259.1.8 ssl_thread_id_callback	1411
5.259.1.9 ssl_thread_stop	1411
5.259.1.10 ssl_verify_privkey	1412

5.259.2 Variable Documentation	1412
5.259.2.1 dh2048	1412
5.259.2.2 dh4096	1412
5.259.2.3 dhparam_lock	1412
5.259.2.4 ecdh1024	1412
5.259.2.5 ecdh512	1412
5.259.2.6 ssl_locks	1412
5.259.2.7 ssl_version	1413
5.260src/providers/cryptography/parameters.c File Reference	1414
5.260.1 Function Documentation	1414
5.260.1.1 dh_exchange_2048	1414
5.260.1.2 dh_exchange_4096	1415
5.260.1.3 dh_params_2048	1415
5.260.1.4 dh_params_4096	1415
5.260.1.5 dh_params_generate	1415
5.260.1.6 dh_params_generate_callback	1415
5.260.1.7 dh_static_2048	1416
5.260.1.8 dh_static_4096	1416
5.260.2 Variable Documentation	1416
5.260.2.1 dh2048	1416
5.260.2.2 dh4096	1416
5.260.2.3 dhparam_lock	1416
5.261src/providers/cryptography/random.c File Reference	1417
5.261.1 Function Documentation	1417
5.261.1.1 rand_choices	1417
5.261.1.2 rand_get_int16	1418
5.261.1.3 rand_get_int32	1418
5.261.1.4 rand_get_int64	1418
5.261.1.5 rand_get_int8	1418
5.261.1.6 rand_get_uint16	1418
5.261.1.7 rand_get_uint32	1419
5.261.1.8 rand_get_uint64	1419
5.261.1.9 rand_get_uint8	1419
5.261.1.10rand_start	1420
5.261.1.11rand_stop	1420
5.261.1.12rand_thread_start	1420
5.261.1.13rand_write	1420
5.261.2 Variable Documentation	1421
5.261.2.1 rand_ctx	1421

5.262src/providers/cryptography/scramble.c File Reference	1422
5.262.1 Function Documentation	1422
5.262.1.1 deprecated_scramble_alloc	1422
5.262.1.2 deprecated_scramble_body_data	1423
5.262.1.3 deprecated_scramble_body_hash	1423
5.262.1.4 deprecated_scramble_body_length	1423
5.262.1.5 deprecated_scramble_cleanup	1424
5.262.1.6 deprecated_scramble_decrypt	1424
5.262.1.7 deprecated_scramble_encrypt	1424
5.262.1.8 deprecated_scramble_free	1425
5.262.1.9 deprecated_scramble_import	1425
5.262.1.10 deprecated_scramble_orig_length	1425
5.262.1.11 deprecated_scramble_total_length	1425
5.262.1.12 deprecated_scramble_vector_data	1426
5.262.1.13 deprecated_scramble_vector_length	1426
5.263src/providers/deprecated/scramble.c File Reference	1427
5.263.1 Function Documentation	1427
5.263.1.1 scramble_alloc	1427
5.263.1.2 scramble_body_data	1428
5.263.1.3 scramble_body_hash	1428
5.263.1.4 scramble_body_length	1428
5.263.1.5 scramble_cleanup	1429
5.263.1.6 scramble_decrypt	1429
5.263.1.7 scramble_encrypt	1429
5.263.1.8 scramble_free	1430
5.263.1.9 scramble_import	1430
5.263.1.10 scramble_orig_length	1430
5.263.1.11 scramble_total_length	1430
5.263.1.12 scramble_vector_data	1431
5.263.1.13 scramble_vector_length	1431
5.264src/providers/cryptography/symmetric.c File Reference	1432
5.264.1 Function Documentation	1432
5.264.1.1 deprecated_symmetric_decrypt	1432
5.264.1.2 deprecated_symmetric_encrypt	1432
5.264.1.3 deprecated_symmetric_key	1433
5.264.1.4 deprecated_symmetric_vector	1433
5.265src/providers/deprecated/symmetric.c File Reference	1434
5.265.1 Function Documentation	1434
5.265.1.1 symmetric_decrypt	1434

5.265.1.2 symmetric_encrypt	1434
5.265.1.3 symmetric_key	1435
5.265.1.4 symmetric_vector	1435
5.266src/providers/cryptography/tls.c File Reference	1436
5.266.1 Function Documentation	1437
5.266.1.1 tls_bits	1437
5.266.1.2 tls_cipher	1437
5.266.1.3 tls_client_alloc	1437
5.266.1.4 tls_continue	1438
5.266.1.5 tls_error	1438
5.266.1.6 tls_free	1438
5.266.1.7 tls_print	1438
5.266.1.8 tls_read	1438
5.266.1.9 tls_server_alloc	1439
5.266.1.10tls_server_create	1439
5.266.1.11tls_server_destroy	1440
5.266.1.12tls_status	1440
5.266.1.13tls_suite	1440
5.266.1.14tls_version	1441
5.266.1.15tls_write	1441
5.267src/providers/database/database.h File Reference	1442
5.267.1 Define Documentation	1445
5.267.1.1 ISNULL	1445
5.267.2 Typedef Documentation	1445
5.267.2.1 row_t	1445
5.267.2.2 table_t	1445
5.267.3 Function Documentation	1446
5.267.3.1 lib_load_mysql	1446
5.267.3.2 lib_version_mysql	1446
5.267.3.3 res_bind_create	1446
5.267.3.4 res_bind_free	1446
5.267.3.5 res_field_block	1447
5.267.3.6 res_field_bool	1447
5.267.3.7 res_field_count	1447
5.267.3.8 res_field_double	1448
5.267.3.9 res_field_float	1448
5.267.3.10res_field_generic	1448
5.267.3.11res_field_int16	1448
5.267.3.12res_field_int32	1449

5.267.3.13	res_field_int64	1449
5.267.3.14	res_field_int8	1449
5.267.3.15	res_field_length	1449
5.267.3.16	res_field_string	1450
5.267.3.17	res_field_uint16	1450
5.267.3.18	res_field_uint32	1450
5.267.3.19	res_field_uint64	1451
5.267.3.20	res_field_uint8	1451
5.267.3.21	res_row_count	1452
5.267.3.22	res_row_get	1452
5.267.3.23	res_row_next	1452
5.267.3.24	res_row_set	1453
5.267.3.25	res_row_store	1453
5.267.3.26	res_stmt_store	1453
5.267.3.27	res_table_alloc	1453
5.267.3.28	res_table_free	1453
5.267.3.29	serv_charset_mysql	1454
5.267.3.30	serv_schema_mysql	1454
5.267.3.31	serv_type_mysql	1454
5.267.3.32	serv_version_mysql	1454
5.267.3.33	sql_errno	1454
5.267.3.34	sql_error	1455
5.267.3.35	sql_insert	1455
5.267.3.36	sql_insert_conn	1455
5.267.3.37	sql_num_rows	1455
5.267.3.38	sql_num_rows_conn	1455
5.267.3.39	sql_open	1456
5.267.3.40	sql_ping	1456
5.267.3.41	sql_query	1456
5.267.3.42	sql_query_conn	1457
5.267.3.43	sql_query_res	1457
5.267.3.44	sql_query_res_conn	1457
5.267.3.45	sql_start	1457
5.267.3.46	sql_stop	1458
5.267.3.47	sql_thread_start	1458
5.267.3.48	sql_thread_stop	1458
5.267.3.49	sql_write	1458
5.267.3.50	sql_write_conn	1459
5.267.3.51	stmt_bind_param	1459

5.267.3.52	stmt_close	1459
5.267.3.53	stmt_errno	1459
5.267.3.54	stmt_error	1459
5.267.3.55	stmt_exec	1460
5.267.3.56	stmt_exec_affected	1460
5.267.3.57	stmt_exec_affected_conn	1460
5.267.3.58	stmt_exec_conn	1461
5.267.3.59	stmt_get_result	1461
5.267.3.60	stmt_get_result_conn	1462
5.267.3.61	stmt_insert	1462
5.267.3.62	stmt_insert_conn	1463
5.267.3.63	stmt_open	1463
5.267.3.64	stmt_prepare	1463
5.267.3.65	stmt_rebuild	1463
5.267.3.66	stmt_reset	1464
5.267.3.67	stmt_start	1464
5.267.3.68	stmt_stop	1464
5.267.3.69	tran_commit	1464
5.267.3.70	tran_rollback	1464
5.267.3.71	tran_start	1465
5.267.4	Variable Documentation	1465
5.267.4.1	sql_pool	1465
5.268	src/providers/database/mysql.c File Reference	1466
5.268.1	Function Documentation	1467
5.268.1.1	lib_load_mysql	1467
5.268.1.2	lib_version_mysql	1467
5.268.1.3	serv_charset_mysql	1467
5.268.1.4	serv_schema_mysql	1467
5.268.1.5	serv_type_mysql	1468
5.268.1.6	serv_version_mysql	1468
5.268.1.7	sql_errno	1468
5.268.1.8	sql_error	1468
5.268.1.9	sql_open	1469
5.268.1.10	sql_ping	1469
5.268.1.11	sql_start	1469
5.268.1.12	sql_stop	1470
5.268.1.13	sql_thread_start	1470
5.268.1.14	sql_thread_stop	1470
5.268.2	Variable Documentation	1470

5.268.2.1 dash	1470
5.268.2.2 lib_version	1471
5.268.2.3 serv_charset	1471
5.268.2.4 serv_schema	1471
5.268.2.5 serv_version	1471
5.268.2.6 sql	1471
5.268.2.7 sql_pool	1471
5.268.2.8 type_embed	1471
5.268.2.9 type_serv	1471
5.269src/providers/database/query.c File Reference	1472
5.269.1 Function Documentation	1472
5.269.1.1 sql_insert	1472
5.269.1.2 sql_insert_conn	1472
5.269.1.3 sql_num_rows	1472
5.269.1.4 sql_num_rows_conn	1472
5.269.1.5 sql_query	1473
5.269.1.6 sql_query_conn	1473
5.269.1.7 sql_query_res	1473
5.269.1.8 sql_query_res_conn	1473
5.269.1.9 sql_write	1474
5.269.1.10sql_write_conn	1474
5.270src/providers/database/results.c File Reference	1475
5.270.1 Function Documentation	1476
5.270.1.1 res_bind_create	1476
5.270.1.2 res_bind_free	1476
5.270.1.3 res_field_block	1477
5.270.1.4 res_field_bool	1477
5.270.1.5 res_field_count	1477
5.270.1.6 res_field_double	1478
5.270.1.7 res_field_float	1478
5.270.1.8 res_field_generic	1478
5.270.1.9 res_field_int16	1478
5.270.1.10res_field_int32	1479
5.270.1.11res_field_int64	1479
5.270.1.12res_field_int8	1479
5.270.1.13res_field_length	1479
5.270.1.14res_field_string	1480
5.270.1.15res_field_uint16	1480
5.270.1.16res_field_uint32	1480

5.270.1.17	res_field_uint64	1481
5.270.1.18	res_field_uint8	1481
5.270.1.19	res_row_count	1482
5.270.1.20	res_row_get	1482
5.270.1.21	res_row_next	1482
5.270.1.22	res_row_set	1483
5.270.1.23	res_row_store	1483
5.270.1.24	res_stmt_store	1483
5.270.1.25	res_table_alloc	1483
5.270.1.26	res_table_free	1483
5.271	src/providers/database/stmts.c File Reference	1485
5.271.1	Function Documentation	1486
5.271.1.1	stmt_bind_param	1486
5.271.1.2	stmt_close	1486
5.271.1.3	stmt_errno	1486
5.271.1.4	stmt_error	1486
5.271.1.5	stmt_exec	1486
5.271.1.6	stmt_exec_affected	1487
5.271.1.7	stmt_exec_affected_conn	1487
5.271.1.8	stmt_exec_conn	1488
5.271.1.9	stmt_get_result	1488
5.271.1.10	stmt_get_result_conn	1489
5.271.1.11	stmt_insert	1489
5.271.1.12	stmt_insert_conn	1489
5.271.1.13	stmt_open	1490
5.271.1.14	stmt_prepare	1490
5.271.1.15	stmt_rebuild	1490
5.271.1.16	stmt_reset	1490
5.271.1.17	stmt_start	1491
5.271.1.18	stmt_stop	1491
5.271.2	Variable Documentation	1491
5.271.2.1	queries	1491
5.272	src/providers/database/transaction.c File Reference	1492
5.272.1	Function Documentation	1492
5.272.1.1	tran_commit	1492
5.272.1.2	tran_rollback	1492
5.272.1.3	tran_start	1493
5.272.2	Variable Documentation	1493
5.272.2.1	command	1493

5.272.2.2 length	1493
5.272.2.3 tran_commands	1493
5.273src/providers/deprecated/deprecated.h File Reference	1494
5.273.1 Function Documentation	1496
5.273.1.1 cryptex_alloc	1496
5.273.1.2 cryptex_body_data	1496
5.273.1.3 cryptex_body_length	1497
5.273.1.4 cryptex_envelope_data	1497
5.273.1.5 cryptex_envelope_length	1497
5.273.1.6 cryptex_free	1497
5.273.1.7 cryptex_hmac_data	1498
5.273.1.8 cryptex_hmac_length	1498
5.273.1.9 cryptex_original_length	1498
5.273.1.10cryptex_total_length	1499
5.273.1.11ecies_decrypt	1499
5.273.1.12ecies_encrypt	1499
5.273.1.13ecies_envelope_derivation	1500
5.273.1.14ecies_group	1500
5.273.1.15ecies_key_alloc	1500
5.273.1.16ecies_key_create	1500
5.273.1.17ecies_key_free	1501
5.273.1.18ecies_key_private	1501
5.273.1.19ecies_key_private_bin	1501
5.273.1.20ecies_key_private_hex	1501
5.273.1.21ecies_key_public	1502
5.273.1.22ecies_key_public_bin	1502
5.273.1.23ecies_key_public_hex	1502
5.273.1.24ecies_start	1502
5.273.1.25ecies_stop	1502
5.273.1.26hmac_digest	1503
5.273.1.27hmac_md4	1503
5.273.1.28hmac_md5	1503
5.273.1.29hmac_ripemd160	1503
5.273.1.30hmac_sha	1503
5.273.1.31hmac_sha1	1503
5.273.1.32hmac_sha224	1504
5.273.1.33hmac_sha256	1504
5.273.1.34hmac_sha384	1504
5.273.1.35hmac_sha512	1504

5.273.1.36	scramble_alloc	1504
5.273.1.37	scramble_body_data	1504
5.273.1.38	scramble_body_hash	1505
5.273.1.39	scramble_body_length	1505
5.273.1.40	scramble_cleanup	1505
5.273.1.41	scramble_decrypt	1505
5.273.1.42	scramble_encrypt	1506
5.273.1.43	scramble_free	1506
5.273.1.44	scramble_import	1506
5.273.1.45	scramble_orig_length	1507
5.273.1.46	scramble_total_length	1507
5.273.1.47	scramble_vector_data	1507
5.273.1.48	scramble_vector_length	1508
5.273.1.49	symmetric_decrypt	1508
5.273.1.50	symmetric_encrypt	1508
5.273.1.51	symmetric_key	1508
5.273.1.52	symmetric_vector	1509
5.274	src/providers/dime/common/crypto_pub.c File Reference	1510
5.274.1	Function Documentation	1510
5.274.1.1	compute_aes256_kek	1510
5.274.1.2	crypto_init	1510
5.274.1.3	crypto_shutdown	1511
5.274.1.4	decrypt_aes_256	1511
5.274.1.5	deserialize_ec_privkey	1511
5.274.1.6	deserialize_ec_pubkey	1511
5.274.1.7	deserialize_ed25519_privkey	1511
5.274.1.8	deserialize_ed25519_pubkey	1511
5.274.1.9	ec_sign_data	1511
5.274.1.10	ec_sign_sha_data	1511
5.274.1.11	ecies_env_derivation	1511
5.274.1.12	ed25519_sign_data	1512
5.274.1.13	ed25519_verify_sig	1512
5.274.1.14	encrypt_aes_256	1512
5.274.1.15	free_ec_key	1512
5.274.1.16	free_ed25519_key	1512
5.274.1.17	free_ed25519_key_chain	1512
5.274.1.18	generate_ec_keypair	1512
5.274.1.19	generate_ed25519_keypair	1512
5.274.1.20	get_random_bytes	1512

5.274.1.2load_ec_privkey	1513
5.274.1.2load_ec_pubkey	1513
5.274.1.23load_ed25519_privkey	1513
5.274.1.24serialize_ec_privkey	1513
5.274.1.25serialize_ec_pubkey	1513
5.274.1.26verify_ec_sha_signature	1513
5.274.1.27verify_ec_signature	1513
5.275src/providers/dime/common/dcrypto.h File Reference	1514
5.275.1 Define Documentation	1515
5.275.1.1 AES_256_KEK_SIZE	1515
5.275.1.2 AES_256_KEY_SIZE	1515
5.275.1.3 AES_256_PADDING_SIZE	1515
5.275.1.4 EC_ENCRYPT_CURVE	1515
5.275.1.5 EC_PUBKEY_SIZE	1515
5.275.1.6 EC_SIGNING_CURVE	1515
5.275.1.7 ED25519_KEY_B64_SIZE	1515
5.275.1.8 ED25519_KEY_SIZE	1516
5.275.1.9 ED25519_SIG_B64_SIZE	1516
5.275.1.10ED25519_SIG_SIZE	1516
5.275.2 Function Documentation	1518
5.275.2.1 PUBLIC_FUNC_DECL	1518
5.275.2.2 PUBLIC_FUNC_DECL	1518
5.275.2.3 PUBLIC_FUNC_DECL	1518
5.275.2.4 PUBLIC_FUNC_DECL	1518
5.275.2.5 PUBLIC_FUNC_DECL	1518
5.275.2.6 PUBLIC_FUNC_DECL	1518
5.275.2.7 PUBLIC_FUNC_DECL	1518
5.275.2.8 PUBLIC_FUNC_DECL	1518
5.275.2.9 PUBLIC_FUNC_DECL	1518
5.275.2.10PUBLIC_FUNC_DECL	1518
5.275.2.11PUBLIC_FUNC_DECL	1518
5.275.2.12PUBLIC_FUNC_DECL	1518
5.275.2.13PUBLIC_FUNC_DECL	1518
5.275.2.14PUBLIC_FUNC_DECL	1518
5.275.2.15PUBLIC_FUNC_DECL	1518
5.275.2.16PUBLIC_FUNC_DECL	1518
5.275.2.17PUBLIC_FUNC_DECL	1518
5.275.2.18PUBLIC_FUNC_DECL	1518
5.275.2.19PUBLIC_FUNC_DECL	1518

5.275.2.20	PUBLIC_FUNC_DECL	1518
5.275.2.21	PUBLIC_FUNC_DECL	1518
5.275.2.22	PUBLIC_FUNC_DECL	1518
5.275.2.23	PUBLIC_FUNC_DECL	1518
5.275.2.24	PUBLIC_FUNC_DECL	1518
5.275.2.25	PUBLIC_FUNC_DECL	1518
5.275.2.26	PUBLIC_FUNC_DECL	1518
5.275.2.27	PUBLIC_FUNC_DECL	1518
5.276	src/providers/dime/common/error.c File Reference	1519
5.276.1	Function Documentation	1520
5.276.1.1	__attribute__	1520
5.276.1.2	_clear_error_stack	1520
5.276.1.3	_create_new_error	1520
5.276.1.4	_dump_error	1520
5.276.1.5	_push_error_stack	1521
5.276.1.6	_push_error_stack_fmt	1521
5.276.1.7	_push_error_stack_openssl	1521
5.276.1.8	_push_error_stack_resolver	1521
5.276.1.9	_push_error_stack_syscall	1521
5.276.1.10	dump_error_stack	1522
5.276.1.11	ldump_last_error	1522
5.276.1.12	get_error_string	1522
5.276.1.13	get_first_error	1522
5.276.1.14	get_last_error	1522
5.276.1.15	get_last_error_code	1522
5.276.1.16	pop_last_error	1523
5.276.2	Variable Documentation	1523
5.276.2.1	_t_err_stack	1523
5.277	src/providers/dime/common/error.h File Reference	1524
5.277.1	Define Documentation	1526
5.277.1.1	ERR_BAD_PARAM	1526
5.277.1.2	ERR_CORRUPTION	1526
5.277.1.3	ERR_NOMEM	1526
5.277.1.4	ERR_OPENSSL	1527
5.277.1.5	ERR_PERM	1527
5.277.1.6	ERR_RESOLVER	1527
5.277.1.7	ERR_STACK_SIZE	1527
5.277.1.8	ERR_SYSCALL	1527
5.277.1.9	ERR_UNSPEC	1527

5.277.1.10	PUBLIC_FUNC_DECL	1528
5.277.1.11	PUBLIC_FUNC_DECL_VA	1528
5.277.1.12	PUBLIC_FUNC_IMPL	1528
5.277.1.13	PUBLIC_FUNC_IMPL_VA1	1529
5.277.1.14	PUBLIC_FUNC_IMPL_VA1_RET	1529
5.277.1.15	PUBLIC_FUNC_IMPL_VA2	1529
5.277.1.16	PUBLIC_FUNC_IMPL_VA2_RET	1529
5.277.1.17	PUBLIC_FUNC_IMPL_VOID	1529
5.277.1.18	PUBLIC_FUNC_PROLOGUE	1529
5.277.1.19	PUBLIC_FUNCTION_IMPLEMENT	1529
5.277.1.20	PUBLIC_FUNCTION_IMPLEMENT_VOID	1530
5.277.1.21	PUSH_ERROR	1530
5.277.1.22	PUSH_ERROR_FMT	1530
5.277.1.23	PUSH_ERROR_OPENSSL	1530
5.277.1.24	PUSH_ERROR_RESOLVER	1531
5.277.1.25	PUSH_ERROR_SYSCALL	1531
5.277.1.26	RET_ERROR_CUST	1531
5.277.1.27	RET_ERROR_CUST_FMT	1531
5.277.1.28	RET_ERROR_INT	1531
5.277.1.29	RET_ERROR_INT_FMT	1532
5.277.1.30	RET_ERROR_PTR	1532
5.277.1.31	RET_ERROR_PTR_FMT	1532
5.277.1.32	RET_ERROR_UINT	1532
5.277.1.33	RET_ERROR_UINT_FMT	1532
5.277.2	Typedef Documentation	1533
5.277.2.1	errinfo_t	1533
5.277.3	Function Documentation	1533
5.277.3.1	__attribute__	1533
5.277.3.2	_clear_error_stack	1533
5.277.3.3	_create_new_error	1533
5.277.3.4	_dump_error	1533
5.277.3.5	_push_error_stack	1533
5.277.3.6	_push_error_stack_fmt	1534
5.277.3.7	_push_error_stack_openssl	1534
5.277.3.8	_push_error_stack_resolver	1534
5.277.3.9	_push_error_stack_syscall	1534
5.277.3.10	dump_error_stack	1534
5.277.3.11	ldump_last_error	1534
5.277.3.12	get_error_string	1535

5.277.3.13	<code>get_first_error</code>	1535
5.277.3.14	<code>get_last_error</code>	1535
5.277.3.15	<code>get_last_error_code</code>	1535
5.277.3.16	<code>pop_last_error</code>	1535
5.278	<code>src/providers/dime/common/misc.c File Reference</code>	1536
5.278.1	Function Documentation	1538
5.278.1.1	<code>__dbgprint</code>	1538
5.278.1.2	<code>__str_printf</code>	1538
5.278.1.3	<code>_b64decode</code>	1539
5.278.1.4	<code>_b64decode_nopad</code>	1539
5.278.1.5	<code>_b64encode</code>	1539
5.278.1.6	<code>_b64encode_nopad</code>	1540
5.278.1.7	<code>_b64encode_w_line separators</code>	1540
5.278.1.8	<code>_compute_crc24_checksum</code>	1540
5.278.1.9	<code>_compute_sha_hash</code>	1540
5.278.1.10	<code>_compute_sha_hash_multibuf</code>	1541
5.278.1.11	<code>_count_ptr_chain</code>	1541
5.278.1.12	<code>_dbgprint</code>	1541
5.278.1.13	<code>_decode_rsa_pubkey</code>	1542
5.278.1.14	<code>_dump_buf</code>	1542
5.278.1.15	<code>_dump_buf_outer</code>	1542
5.278.1.16	<code>_encode_rsa_pubkey</code>	1543
5.278.1.17	<code>_get_chr_date</code>	1543
5.278.1.18	<code>_get_dbg_level</code>	1544
5.278.1.19	<code>_get_x509_cert_sha_hash</code>	1544
5.278.1.20	<code>_hex_encode</code>	1544
5.278.1.21	<code>_int_no_get_2b</code>	1544
5.278.1.22	<code>_int_no_get_3b</code>	1545
5.278.1.23	<code>_int_no_get_4b</code>	1545
5.278.1.24	<code>_int_no_put_2b</code>	1545
5.278.1.25	<code>_int_no_put_3b</code>	1545
5.278.1.26	<code>_int_no_put_4b</code>	1546
5.278.1.27	<code>_is_buf_zeroed</code>	1546
5.278.1.28	<code>_mem_append</code>	1546
5.278.1.29	<code>_ptr_chain_add</code>	1547
5.278.1.30	<code>_ptr_chain_clone</code>	1547
5.278.1.31	<code>_ptr_chain_free</code>	1547
5.278.1.32	<code>_read_file_data</code>	1547
5.278.1.33	<code>_read_pem_data</code>	1548

5.278.1.34_secure_wipe	1548
5.278.1.35_set_dbg_level	1548
5.278.1.36_str_printf	1549
5.278.1.37_write_pem_data	1549
5.278.2 Variable Documentation	1549
5.278.2.1 _verbose	1549
5.279src/providers/dime/common/misc.h File Reference	1550
5.279.1 Define Documentation	1551
5.279.1.1 ALERT_PRINT	1551
5.279.1.2 ANSI_COLOR_RED	1551
5.279.1.3 ANSI_COLOR_RESET	1551
5.279.1.4 B64_DECODED_LEN	1551
5.279.1.5 B64_ENCODED_LEN	1552
5.279.1.6 chr_isprint	1552
5.279.1.7 chr_isspace	1552
5.279.1.8 SHA_160_SIZE	1552
5.279.1.9 SHA_256_SIZE	1552
5.279.1.10SHA_512_B64_SIZE	1552
5.279.1.11SHA_512_SIZE	1552
5.279.2 Function Documentation	1554
5.279.2.1 PUBLIC_FUNC_DECL	1554
5.279.2.2 PUBLIC_FUNC_DECL	1554
5.279.2.3 PUBLIC_FUNC_DECL	1554
5.279.2.4 PUBLIC_FUNC_DECL	1554
5.279.2.5 PUBLIC_FUNC_DECL	1554
5.279.2.6 PUBLIC_FUNC_DECL	1554
5.279.2.7 PUBLIC_FUNC_DECL	1554
5.279.2.8 PUBLIC_FUNC_DECL	1554
5.279.2.9 PUBLIC_FUNC_DECL	1554
5.279.2.10PUBLIC_FUNC_DECL	1554
5.279.2.11PUBLIC_FUNC_DECL	1554
5.279.2.12PUBLIC_FUNC_DECL	1554
5.279.2.13PUBLIC_FUNC_DECL	1554
5.279.2.14PUBLIC_FUNC_DECL	1554
5.279.2.15PUBLIC_FUNC_DECL	1554
5.279.2.16PUBLIC_FUNC_DECL	1554
5.279.2.17PUBLIC_FUNC_DECL	1554
5.279.2.18PUBLIC_FUNC_DECL	1554
5.279.2.19PUBLIC_FUNC_DECL	1554

5.279.2.20	PUBLIC_FUNC_DECL	1554
5.279.2.21	PUBLIC_FUNC_DECL	1554
5.279.2.22	PUBLIC_FUNC_DECL	1554
5.279.2.23	PUBLIC_FUNC_DECL	1554
5.279.2.24	PUBLIC_FUNC_DECL	1554
5.279.2.25	PUBLIC_FUNC_DECL	1554
5.279.2.26	PUBLIC_FUNC_DECL	1554
5.279.2.27	PUBLIC_FUNC_DECL	1554
5.279.2.28	PUBLIC_FUNC_DECL	1554
5.279.2.29	PUBLIC_FUNC_DECL	1554
5.279.2.30	PUBLIC_FUNC_DECL	1554
5.279.2.31	PUBLIC_FUNC_DECL	1554
5.279.2.32	PUBLIC_FUNC_DECL	1554
5.279.2.33	PUBLIC_FUNC_DECL	1554
5.279.2.34	PUBLIC_FUNC_DECL_VA	1554
5.279.2.35	PUBLIC_FUNC_DECL_VA	1554
5.279.3	Variable Documentation	1554
5.279.3.1	_verbose	1554
5.280	src/providers/dime/common/misc_pub.c File Reference	1556
5.280.1	Function Documentation	1556
5.280.1.1	b64decode	1556
5.280.1.2	b64decode_nopad	1556
5.280.1.3	b64encode	1557
5.280.1.4	b64encode_nopad	1557
5.280.1.5	compute_sha_hash	1557
5.280.1.6	compute_sha_hash_multibuf	1557
5.280.1.7	count_ptr_chain	1557
5.280.1.8	dbgprint	1557
5.280.1.9	decode_rsa_pubkey	1557
5.280.1.10	dump_buf	1557
5.280.1.11	ldump_buf_outer	1557
5.280.1.12	encode_rsa_pubkey	1558
5.280.1.13	get_chr_date	1558
5.280.1.14	get_dbg_level	1558
5.280.1.15	get_x509_cert_sha_hash	1558
5.280.1.16	hex_encode	1558
5.280.1.17	int_no_get_2b	1558
5.280.1.18	int_no_get_3b	1558
5.280.1.19	int_no_get_4b	1558

5.280.1.20	int_no_put_2b	1558
5.280.1.21	int_no_put_3b	1559
5.280.1.22	int_no_put_4b	1559
5.280.1.23	is_buf_zeroed	1559
5.280.1.24	mem_append	1559
5.280.1.25	ptr_chain_add	1559
5.280.1.26	ptr_chain_clone	1559
5.280.1.27	ptr_chain_free	1559
5.280.1.28	read_file_data	1559
5.280.1.29	read_pem_data	1559
5.280.1.30	secure_wipe	1560
5.280.1.31	set_dbg_level	1560
5.280.1.32	str_printf	1560
5.281	src/providers/dime/common/network.c File Reference	1561
5.281.1	Define Documentation	1561
5.281.1.1	CONNECT_TIMEOUT	1561
5.281.2	Function Documentation	1561
5.281.2.1	_connect_host	1561
5.281.2.2	_connect_timeout	1562
5.282	src/providers/dime/common/network_pub.c File Reference	1563
5.282.1	Function Documentation	1563
5.282.1.1	connect_host	1563
5.283	src/providers/dime/dime_ctx.c File Reference	1564
5.283.1	Function Documentation	1564
5.283.1.1	dime_ctx_free	1564
5.283.1.2	dime_ctx_log	1564
5.283.1.3	dime_ctx_new	1564
5.283.2	Variable Documentation	1564
5.283.2.1	LOG_LEVEL_DEBUG	1564
5.283.2.2	LOG_LEVEL_ERROR	1564
5.283.2.3	LOG_LEVEL_INFO	1565
5.284	src/providers/dime/dime_ctx.h File Reference	1566
5.284.1	Define Documentation	1566
5.284.1.1	DIME_LOG_DEBUG	1566
5.284.1.2	DIME_LOG_ERROR	1567
5.284.1.3	DIME_LOG_INFO	1567
5.284.2	Typedef Documentation	1567
5.284.2.1	dime_ctx_t	1567
5.284.2.2	log_function_t	1567

5.284.3 Enumeration Type Documentation	1567
5.284.3.1 log_code_t	1567
5.284.4 Function Documentation	1567
5.284.4.1 dime_ctx_free	1567
5.284.4.2 dime_ctx_log	1568
5.284.4.3 dime_ctx_new	1568
5.284.5 Variable Documentation	1568
5.284.5.1 LOG_LEVEL_DEBUG	1568
5.284.5.2 LOG_LEVEL_ERROR	1568
5.284.5.3 LOG_LEVEL_INFO	1568
5.285src/providers/dime/dmessage/common.h File Reference	1569
5.285.1 Define Documentation	1569
5.285.1.1 ALTERNATE_PADDING_ALGORITHM_ENABLED	1569
5.285.1.2 ALTERNATE_USER_KEY_APPLIED_TO_DATE	1570
5.285.1.3 CHUNK_HEADER_SIZE	1570
5.285.1.4 CHUNK_LENGTH_SIZE	1570
5.285.1.5 DATA_SEGMENT_CONTINUATION_ENABLED	1570
5.285.1.6 DEFAULT_CHUNK_FLAGS	1570
5.285.1.7 DMIME_CHUNK_TYPE_MAX	1570
5.285.1.8 DMIME_NUM_COMMON_HEADERS	1570
5.285.1.9 GZIP_COMPRESSION_ENABLED	1570
5.285.1.10MESSAGE_HEADER_SIZE	1570
5.285.1.11MESSAGE_LENGTH_SIZE	1570
5.285.1.12MINIMUM_PAYLOAD_SIZE	1570
5.285.1.13TRACING_HEADER_SIZE	1570
5.285.2 Enumeration Type Documentation	1571
5.285.2.1 dmime_bounce_type_t	1571
5.285.2.2 dmime_chunk_section_t	1571
5.285.2.3 dmime_chunk_type_t	1571
5.285.2.4 dmime_payload_type_t	1572
5.285.3 Variable Documentation	1572
5.285.3.1 dmime_chunk_keys	1572
5.286src/providers/dime/signet/common.h File Reference	1573
5.286.1 Define Documentation	1574
5.286.1.1 DIME_NUMBER_SIZE	1574
5.286.1.2 FIELD_NAME_MAX_SIZE	1574
5.286.1.3 KEYS_FID_MAX	1574
5.286.1.4 KEYS_HEADER_SIZE	1574
5.286.1.5 SIGNET_FID_MAX	1575

5.286.1.6	SIGNET_HEADER_SIZE	1575
5.286.1.7	SIGNET_KEY_ORG	1575
5.286.1.8	SIGNET_KEY_USER	1575
5.286.1.9	SIGNET_MAX_SIZE	1575
5.286.1.10	SIGNET_ORG	1575
5.286.1.11	SIGNET_USER	1575
5.286.1.12	SIGNET_VER_NO	1575
5.286.1.13	UNSIGNED_MAX_1_BYTE	1575
5.286.1.14	UNSIGNED_MAX_2_BYTE	1575
5.286.1.15	UNSIGNED_MAX_3_BYTE	1575
5.286.2	Enumeration Type Documentation	1576
5.286.2.1	dime_number_t	1576
5.286.2.2	field_data_t	1576
5.286.2.3	keys_org_t	1576
5.286.2.4	keys_user_t	1576
5.286.2.5	signet_org_field_t	1577
5.286.2.6	signet_ssr_field_t	1577
5.286.2.7	signet_user_field_t	1578
5.286.2.8	signkey_format_t	1578
5.286.2.9	sok_permissions_t	1579
5.286.3	Function Documentation	1579
5.286.3.1	dime_number_to_str	1579
5.286.4	Variable Documentation	1579
5.286.4.1	signet_org_field_keys	1579
5.286.4.2	signet_ssr_field_keys	1579
5.286.4.3	signet_user_field_keys	1579
5.287	src/providers/dime/dmessage/crypto.h File Reference	1580
5.287.1	Define Documentation	1582
5.287.1.1	TRACING_LENGTH_SIZE	1582
5.287.2	Typedef Documentation	1582
5.287.2.1	dmime_object_chunk_t	1582
5.287.3	Enumeration Type Documentation	1582
5.287.3.1	dmime_actor_t	1582
5.287.3.2	dmime_message_chunk_state_t	1582
5.287.3.3	dmime_message_state_t	1582
5.287.3.4	dmime_object_state_t	1583
5.287.4	Function Documentation	1583
5.287.4.1	__attribute__	1583
5.287.4.2	dime_dmsg_actor_to_string	1583

5.287.4.3	dime_dmsg_chunks_sig_origin_sign	1583
5.287.4.4	dime_dmsg_kek_in_derive	1584
5.287.4.5	dime_dmsg_message_binary_deserialize	1584
5.287.4.6	dime_dmsg_message_binary_serialize	1584
5.287.4.7	dime_dmsg_message_decrypt_as_auth	1585
5.287.4.8	dime_dmsg_message_decrypt_as_dest	1585
5.287.4.9	dime_dmsg_message_decrypt_as_orig	1585
5.287.4.10	dime_dmsg_message_decrypt_as_recip	1586
5.287.4.11	dime_dmsg_message_destroy	1586
5.287.4.12	dime_dmsg_message_encrypt	1586
5.287.4.13	dime_dmsg_message_envelope_decrypt	1586
5.287.4.14	dime_dmsg_message_state_get	1587
5.287.4.15	dime_dmsg_object_chunk_create	1587
5.287.4.16	dime_dmsg_object_chunklist_destroy	1587
5.287.4.17	dime_dmsg_object_destroy	1588
5.287.4.18	dime_dmsg_object_dump	1588
5.287.4.19	dime_dmsg_object_state_init	1588
5.287.4.20	dime_dmsg_object_state_to_string	1588
5.287.5	Variable Documentation	1589
5.287.5.1	dmime_kek_t	1589
5.287.5.2	dmime_message_chunk_t	1589
5.287.5.3	dmime_tracing_t	1589
5.288	src/providers/dime/dmessage/dmsg.c File Reference	1590
5.288.1	Define Documentation	1591
5.288.1.1	CKEY_EMPTY	1591
5.288.2	Typedef Documentation	1591
5.288.2.1	dmime_encrypted_payload_t	1591
5.288.2.2	dmime_ephemeral_payload_t	1592
5.288.2.3	dmime_kekset_t	1592
5.288.2.4	dmime_signature_payload_t	1592
5.288.3	Function Documentation	1592
5.288.3.1	__attribute__	1592
5.288.3.2	dime_dmsg_actor_to_string	1592
5.288.3.3	dime_dmsg_chunks_sig_origin_sign	1592
5.288.3.4	dime_dmsg_kek_in_derive	1593
5.288.3.5	dime_dmsg_message_binary_deserialize	1593
5.288.3.6	dime_dmsg_message_binary_serialize	1593
5.288.3.7	dime_dmsg_message_decrypt_as_auth	1593
5.288.3.8	dime_dmsg_message_decrypt_as_dest	1594

5.288.3.9	dime_dmsg_message_decrypt_as_orig	1594
5.288.3.10	dime_dmsg_message_decrypt_as_recip	1594
5.288.3.11	dime_dmsg_message_destroy	1595
5.288.3.12	dime_dmsg_message_encrypt	1595
5.288.3.13	dime_dmsg_message_envelope_decrypt	1595
5.288.3.14	dime_dmsg_message_state_get	1596
5.288.3.15	dime_dmsg_object_chunk_create	1596
5.288.3.16	dime_dmsg_object_chunklist_destroy	1596
5.288.3.17	dime_dmsg_object_destroy	1596
5.288.3.18	dime_dmsg_object_dump	1597
5.288.3.19	dime_dmsg_object_state_init	1597
5.288.3.20	dime_dmsg_object_state_to_string	1597
5.288.4	Variable Documentation	1597
5.288.4.1	dmime_chunk_keys	1597
5.288.4.2	dmime_keyslot_t	1597
5.288.4.3	dmime_standard_payload_t	1598
5.289	src/providers/dime/dmessage/parse.h File Reference	1599
5.289.1	Enumeration Type Documentation	1599
5.289.1.1	dmime_header_type_t	1599
5.289.2	Function Documentation	1600
5.289.2.1	dime_prsr_envelope_destroy	1600
5.289.2.2	dime_prsr_envelope_format	1600
5.289.2.3	dime_prsr_envelope_parse	1600
5.289.2.4	dime_prsr_headers_create	1601
5.289.2.5	dime_prsr_headers_destroy	1601
5.289.2.6	dime_prsr_headers_format	1601
5.289.2.7	dime_prsr_headers_parse	1601
5.289.3	Variable Documentation	1602
5.289.3.1	dmime_header_keys	1602
5.290	src/providers/dime/dmessage/parser.c File Reference	1603
5.290.1	Function Documentation	1603
5.290.1.1	dime_prsr_envelope_destroy	1603
5.290.1.2	dime_prsr_envelope_format	1603
5.290.1.3	dime_prsr_envelope_parse	1604
5.290.1.4	dime_prsr_headers_create	1604
5.290.1.5	dime_prsr_headers_destroy	1604
5.290.1.6	dime_prsr_headers_format	1605
5.290.1.7	dime_prsr_headers_parse	1605
5.290.2	Variable Documentation	1605

5.290.2.1	dmime_header_keys	1605
5.291	src/providers/dime/dmtp/commands.c File Reference	1606
5.291.1	Define Documentation	1607
5.291.1.1	DMTP_EMPTY_ARG	1607
5.291.1.2	DMTP_IGNORE_ARG	1607
5.291.2	Function Documentation	1607
5.291.2.1	dime_dmtp_command_create	1607
5.291.2.2	dime_dmtp_command_data	1608
5.291.2.3	dime_dmtp_command_destroy	1608
5.291.2.4	dime_dmtp_command_ehlo	1608
5.291.2.5	dime_dmtp_command_format	1608
5.291.2.6	dime_dmtp_command_helo	1609
5.291.2.7	dime_dmtp_command_help	1609
5.291.2.8	dime_dmtp_command_hist	1609
5.291.2.9	dime_dmtp_command_mail	1609
5.291.2.10	dime_dmtp_command_mode	1610
5.291.2.11	dime_dmtp_command_noop	1610
5.291.2.12	dime_dmtp_command_parse	1610
5.291.2.13	dime_dmtp_command_quit	1611
5.291.2.14	dime_dmtp_command_rcpt	1611
5.291.2.15	dime_dmtp_command_rset	1611
5.291.2.16	dime_dmtp_command_sgent_domain	1611
5.291.2.17	dime_dmtp_command_sgent_user	1612
5.291.2.18	dime_dmtp_command_starttls	1612
5.291.2.19	dime_dmtp_command_vrfy_domain	1612
5.291.2.20	dime_dmtp_command_vrfy_user	1612
5.291.3	Variable Documentation	1613
5.291.3.1	dmtp_command_list	1613
5.292	src/servers/dmtp/commands.c File Reference	1614
5.292.1	Function Documentation	1614
5.292.1.1	dmtp_compare	1614
5.292.1.2	dmtp_process	1614
5.292.1.3	dmtp_requeue	1614
5.292.1.4	dmtp_sort	1615
5.293	src/servers/imap/commands.c File Reference	1616
5.293.1	Function Documentation	1616
5.293.1.1	imap_compare	1616
5.293.1.2	imap_process	1616
5.293.1.3	imap_requeue	1617

5.293.1.4 imap_sort	1617
5.294src/servers/molten/commands.c File Reference	1618
5.294.1 Function Documentation	1618
5.294.1.1 molten_compare	1618
5.294.1.2 molten_parse	1618
5.294.1.3 molten_sort	1618
5.295src/servers/pop/commands.c File Reference	1619
5.295.1 Function Documentation	1619
5.295.1.1 pop_compare	1619
5.295.1.2 pop_process	1619
5.295.1.3 pop_requeue	1619
5.295.1.4 pop_sort	1619
5.296src/servers/smtp/commands.c File Reference	1620
5.296.1 Function Documentation	1620
5.296.1.1 smtp_compare	1620
5.296.1.2 smtp_process	1620
5.296.1.3 smtp_requeue	1620
5.296.1.4 smtp_sort	1621
5.297src/providers/dime/dmtp/commands.h File Reference	1622
5.297.1 Define Documentation	1624
5.297.1.1 DMTP_COMMANDS_NUM	1624
5.297.1.2 DMTP_MAX_ARGUMENT_NUM	1624
5.297.2 Enumeration Type Documentation	1624
5.297.2.1 dmtp_argument_type_t	1624
5.297.2.2 dmtp_command_type_t	1624
5.297.2.3 dmtp_mode_t	1625
5.297.2.4 dmtp_parse_state_t	1625
5.297.3 Function Documentation	1625
5.297.3.1 dime_dmtp_command_create	1625
5.297.3.2 dime_dmtp_command_data	1625
5.297.3.3 dime_dmtp_command_destroy	1626
5.297.3.4 dime_dmtp_command_ehlo	1626
5.297.3.5 dime_dmtp_command_format	1626
5.297.3.6 dime_dmtp_command_helo	1626
5.297.3.7 dime_dmtp_command_help	1627
5.297.3.8 dime_dmtp_command_hist	1627
5.297.3.9 dime_dmtp_command_mail	1627
5.297.3.10dime_dmtp_command_mode	1627
5.297.3.11dime_dmtp_command_noop	1628

5.297.3.12	dime_dmtplib_command_parse	1628
5.297.3.13	dime_dmtplib_command_quit	1628
5.297.3.14	dime_dmtplib_command_rcpt	1628
5.297.3.15	dime_dmtplib_command_rset	1629
5.297.3.16	dime_dmtplib_command_sgnt_domain	1629
5.297.3.17	dime_dmtplib_command_sgnt_user	1629
5.297.3.18	dime_dmtplib_command_starttls	1629
5.297.3.19	dime_dmtplib_command_vrfy_domain	1630
5.297.3.20	dime_dmtplib_command_vrfy_user	1630
5.297.4	Variable Documentation	1630
5.297.4.1	dmtplib_command_list	1630
5.298	src/servers/dmtplib/commands.h File Reference	1631
5.298.1	Variable Documentation	1631
5.298.1.1	dmtplib_commands	1631
5.299	src/servers/http/commands.h File Reference	1632
5.300	src/servers/imap/commands.h File Reference	1633
5.300.1	Variable Documentation	1633
5.300.1.1	imap_commands	1633
5.301	src/servers/molten/commands.h File Reference	1634
5.301.1	Variable Documentation	1634
5.301.1.1	molten_commands	1634
5.302	src/servers/pop/commands.h File Reference	1635
5.302.1	Variable Documentation	1635
5.302.1.1	pop_commands	1635
5.303	src/servers/smtp/commands.h File Reference	1636
5.303.1	Variable Documentation	1636
5.303.1.1	smtp_commands	1636
5.304	src/providers/dime/ed25519/curve25519-donna-32bit.h File Reference	1637
5.304.1	Define Documentation	1637
5.304.1.1	carry_pass	1637
5.304.1.2	carry_pass_final	1637
5.304.1.3	carry_pass_full	1637
5.304.1.4	curve25519_mul_noinline	1637
5.304.1.5	F	1638
5.304.1.6	F	1638
5.304.2	Typedef Documentation	1638
5.304.2.1	bignum25519	1638
5.304.2.2	bignum25519align16	1638
5.305	src/providers/dime/ed25519/curve25519-donna-64bit.h File Reference	1639

5.305.1 Define Documentation	1639
5.305.1.1 curve25519_contract_carry	1639
5.305.1.2 curve25519_contract_carry_final	1639
5.305.1.3 curve25519_contract_carry_full	1639
5.305.1.4 ED25519_64BIT_TABLES	1639
5.305.1.5 F	1639
5.305.1.6 write51	1640
5.305.1.7 write51full	1640
5.305.2 Typedef Documentation	1640
5.305.2.1 bignum25519	1640
5.306src/providers/dime/ed25519/curve25519-donna-helpers.h File Reference	1641
5.307src/providers/dime/ed25519/curve25519-donna-sse2.h File Reference	1642
5.307.1 Define Documentation	1642
5.307.1.1 carry_pass	1642
5.307.1.2 carry_pass_final	1642
5.307.1.3 carry_pass_full	1643
5.307.1.4 curve25519_add_after_basic	1643
5.307.1.5 curve25519_square	1643
5.307.1.6 F	1643
5.307.2 Typedef Documentation	1643
5.307.2.1 bignum25519	1643
5.307.2.2 packed32bignum25519	1643
5.307.2.3 packed64bignum25519	1643
5.307.2.4 packedelem32	1643
5.307.2.5 packedelem64	1643
5.307.2.6 packedelem8	1643
5.307.2.7 xmmi	1643
5.308src/providers/dime/ed25519/ed25519-donna-32bit-sse2.h File Reference	1644
5.309src/providers/dime/ed25519/ed25519-donna-32bit-tables.h File Reference	1645
5.310src/providers/dime/ed25519/ed25519-donna-64bit-sse2.h File Reference	1646
5.311src/providers/dime/ed25519/ed25519-donna-64bit-tables.h File Reference	1647
5.312src/providers/dime/ed25519/ed25519-donna-64bit-x86-32bit.h File Reference	1648
5.313src/providers/dime/ed25519/ed25519-donna-64bit-x86.h File Reference	1649
5.314src/providers/dime/ed25519/ed25519-donna-basepoint-table.h File Reference	1650
5.315src/providers/dime/ed25519/ed25519-donna-batchverify.h File Reference	1651
5.315.1 Define Documentation	1651
5.315.1.1 heap_batch_size	1651
5.315.1.2 max_batch_size	1651
5.315.2 Typedef Documentation	1651

5.315.2.1 batch_heap	1651
5.315.2.2 heap_index_t	1651
5.315.3 Function Documentation	1651
5.315.3.1 ed25519_sign_open_batch	1651
5.315.4 Variable Documentation	1652
5.315.4.1 batch_point_buffer	1652
5.316src/providers/dime/ed25519/ed25519-donna-impl-base.h File Reference	1653
5.316.1 Define Documentation	1653
5.316.1.1 S1_SWINDOWSIZE	1653
5.316.1.2 S1_TABLE_SIZE	1653
5.316.1.3 S2_SWINDOWSIZE	1653
5.316.1.4 S2_TABLE_SIZE	1653
5.317src/providers/dime/ed25519/ed25519-donna-impl-sse2.h File Reference	1654
5.317.1 Define Documentation	1654
5.317.1.1 S1_SWINDOWSIZE	1654
5.317.1.2 S1_TABLE_SIZE	1654
5.317.1.3 S2_SWINDOWSIZE	1654
5.317.1.4 S2_TABLE_SIZE	1654
5.318src/providers/dime/ed25519/ed25519-donna-portable-identify.h File Reference	1655
5.318.1 Define Documentation	1655
5.318.1.1 OS_NIX	1655
5.319src/providers/dime/ed25519/ed25519-donna-portable.h File Reference	1656
5.319.1 Define Documentation	1656
5.319.1.1 ALIGN	1656
5.319.1.2 DONNA_INLINE	1656
5.319.1.3 DONNA_NOINLINE	1656
5.319.1.4 mul32x32_64	1656
5.319.1.5 ROTL32	1656
5.319.1.6 ROTR32	1656
5.320src/providers/dime/ed25519/ed25519-donna.h File Reference	1657
5.320.1 Define Documentation	1657
5.320.1.1 ED25519_32BIT	1657
5.320.2 Typedef Documentation	1657
5.320.2.1 ge25519	1657
5.320.2.2 ge25519_niels	1657
5.320.2.3 ge25519_p1p1	1657
5.320.2.4 ge25519_pniels	1657
5.320.2.5 hash_512bits	1657
5.321src/providers/dime/ed25519/fuzz/ed25519-donna.h File Reference	1658

5.321.1 Typedef Documentation	1658
5.321.1.1 <code>curved25519_key</code>	1658
5.321.1.2 <code>ed25519_public_key</code>	1658
5.321.1.3 <code>ed25519_secret_key</code>	1658
5.321.1.4 <code>ed25519_signature</code>	1658
5.321.2 Function Documentation	1659
5.321.2.1 <code>curved25519_scalarmult_basepoint</code>	1659
5.321.2.2 <code>curved25519_scalarmult_basepoint_sse2</code>	1659
5.321.2.3 <code>ed25519_publickey</code>	1659
5.321.2.4 <code>ed25519_publickey_sse2</code>	1659
5.321.2.5 <code>ed25519_randombytes_unsafe</code>	1659
5.321.2.6 <code>ed25519_randombytes_unsafe_sse2</code>	1659
5.321.2.7 <code>ed25519_sign</code>	1659
5.321.2.8 <code>ed25519_sign_open</code>	1659
5.321.2.9 <code>ed25519_sign_open_batch</code>	1660
5.321.2.10 <code>ed25519_sign_open_batch_sse2</code>	1660
5.321.2.11 <code>ed25519_sign_open_sse2</code>	1660
5.321.2.12 <code>ed25519_sign_sse2</code>	1660
5.322 <code>src/providers/dime/ed25519/ed25519-hash-custom.h</code> File Reference	1661
5.323 <code>src/providers/dime/ed25519/ed25519-hash.h</code> File Reference	1662
5.323.1 Typedef Documentation	1662
5.323.1.1 <code>ed25519_hash_context</code>	1662
5.324 <code>src/providers/dime/ed25519/ed25519-randombytes-custom.h</code> File Reference	1663
5.325 <code>src/providers/dime/ed25519/ed25519-randombytes.h</code> File Reference	1664
5.325.1 Function Documentation	1664
5.325.1.1 <code>ed25519_randombytes_unsafe</code>	1664
5.326 <code>src/providers/dime/ed25519/ed25519.c</code> File Reference	1665
5.326.1 Function Documentation	1665
5.326.1.1 <code>curved25519_scalarmult_basepoint</code>	1665
5.326.1.2 <code>ed25519_publickey</code>	1665
5.326.1.3 <code>ed25519_sign</code>	1665
5.326.1.4 <code>ed25519_sign_open</code>	1665
5.327 <code>src/providers/prime/cryptography/ed25519.c</code> File Reference	1666
5.327.1 Function Documentation	1666
5.327.1.1 <code>ed25519_alloc</code>	1666
5.327.1.2 <code>ed25519_free</code>	1666
5.327.1.3 <code>ed25519_generate</code>	1666
5.327.1.4 <code>ed25519_private_get</code>	1667
5.327.1.5 <code>ed25519_private_set</code>	1667

5.327.1.6 ed25519_public_get	1667
5.327.1.7 ed25519_public_set	1667
5.327.1.8 ed25519_sign	1667
5.327.1.9 ed25519_type	1667
5.327.1.10 ed25519_verify	1668
5.328src/providers/dime/ed25519/ed25519.h File Reference	1669
5.328.1 Define Documentation	1669
5.328.1.1 ED25519_FN	1669
5.328.1.2 ED25519_FN2	1669
5.328.1.3 ED25519_FN3	1669
5.328.1.4 ED25519_SUFFIX	1669
5.328.2 Typedef Documentation	1670
5.328.2.1 curved25519_key	1670
5.328.2.2 ed25519_public_key	1670
5.328.2.3 ed25519_secret_key	1670
5.328.2.4 ed25519_signature	1670
5.328.3 Function Documentation	1670
5.328.3.1 curved25519_scalarmult_basepoint_donna	1670
5.328.3.2 ed25519_publickey_donna	1670
5.328.3.3 ed25519_randombytes_unsafe_donna	1670
5.328.3.4 ed25519_sign_donna	1670
5.328.3.5 ed25519_sign_open_batch_donna	1670
5.328.3.6 ed25519_sign_open_donna	1670
5.329src/providers/dime/ed25519/fuzz/curve25519-ref10.c File Reference	1671
5.329.1 Typedef Documentation	1671
5.329.1.1 crypto_int32	1671
5.329.1.2 crypto_int64	1671
5.329.1.3 crypto_uint64	1671
5.329.1.4 fe	1671
5.329.2 Function Documentation	1672
5.329.2.1 crypto_scalarmult_base_ref10	1672
5.329.2.2 crypto_scalarmult_ref10	1672
5.329.2.3 fe_0	1672
5.329.2.4 fe_1	1672
5.329.2.5 fe_add	1672
5.329.2.6 fe_copy	1672
5.329.2.7 fe_cswap	1672
5.329.2.8 fe_frombytes	1672
5.329.2.9 fe_invert	1673

5.329.2.10fe_mul	1673
5.329.2.11fe_mul121666	1673
5.329.2.12fe_sq	1673
5.329.2.13fe_sub	1673
5.329.2.14fe_tobytes	1673
5.330src/providers/dime/ed25519/fuzz/curve25519-ref10.h File Reference	1674
5.330.1 Function Documentation	1674
5.330.1.1 crypto_scalarmult_base_ref10	1674
5.330.1.2 crypto_scalarmult_ref10	1674
5.331src/providers/dime/ed25519/fuzz/ed25519-donna-sse2.c File Reference	1675
5.331.1 Define Documentation	1675
5.331.1.1 ED25519_SSE2	1675
5.331.1.2 ED25519_SUFFIX	1675
5.332src/providers/dime/ed25519/fuzz/ed25519-donna.c File Reference	1676
5.333src/providers/dime/ed25519/fuzz/ed25519-ref10.c File Reference	1677
5.333.1 Define Documentation	1677
5.333.1.1 F	1677
5.333.2 Typedef Documentation	1677
5.333.2.1 crypto_int32	1677
5.333.2.2 crypto_int64	1677
5.333.2.3 crypto_uint32	1678
5.333.2.4 crypto_uint64	1678
5.333.2.5 fe	1678
5.333.3 Function Documentation	1678
5.333.3.1 crypto_sign_open_ref10	1678
5.333.3.2 crypto_sign_pk_ref10	1678
5.333.3.3 crypto_sign_ref10	1678
5.334src/providers/dime/ed25519/fuzz/ed25519-ref10.h File Reference	1679
5.334.1 Function Documentation	1679
5.334.1.1 crypto_sign_open_ref10	1679
5.334.1.2 crypto_sign_pk_ref10	1679
5.334.1.3 crypto_sign_ref10	1679
5.335src/providers/dime/ed25519/fuzz/fuzz-curve25519.c File Reference	1680
5.335.1 Define Documentation	1680
5.335.1.1 quarter	1680
5.335.1.2 rotl32	1680
5.335.2 Function Documentation	1680
5.335.2.1 main	1680
5.335.2.2 prng	1680

5.336src/providers/dime/ed25519/fuzz/fuzz-ed25519.c File Reference	1681
5.336.1 Define Documentation	1681
5.336.1.1 quarter	1681
5.336.1.2 rotl32	1682
5.336.2 Typedef Documentation	1682
5.336.2.1 generated_data	1682
5.336.2.2 random_data	1682
5.336.3 Function Documentation	1682
5.336.3.1 main	1682
5.336.3.2 prng	1682
5.337src/providers/dime/ed25519/modm-donna-32bit.h File Reference	1683
5.337.1 Define Documentation	1683
5.337.1.1 bignum256modm_bits_per_limb	1683
5.337.1.2 bignum256modm_limb_size	1683
5.337.2 Typedef Documentation	1683
5.337.2.1 bignum256modm	1683
5.337.2.2 bignum256modm_element_t	1683
5.338src/providers/dime/ed25519/modm-donna-64bit.h File Reference	1684
5.338.1 Define Documentation	1684
5.338.1.1 bignum256modm_bits_per_limb	1684
5.338.1.2 bignum256modm_limb_size	1684
5.338.2 Typedef Documentation	1684
5.338.2.1 bignum256modm	1684
5.338.2.2 bignum256modm_element_t	1684
5.339src/providers/dime/ed25519/regression.h File Reference	1685
5.340src/providers/dime/ed25519/test-internals.c File Reference	1686
5.340.1 Function Documentation	1686
5.340.1.1 main	1686
5.341src/providers/dime/ed25519/test-ticks.h File Reference	1687
5.341.1 Define Documentation	1687
5.341.1.1 maxticks	1687
5.341.1.2 timeit	1687
5.342src/providers/dime/ed25519/test.c File Reference	1688
5.342.1 Define Documentation	1688
5.342.1.1 test_batch_count	1688
5.342.1.2 test_batch_rounds	1688
5.342.2 Typedef Documentation	1689
5.342.2.1 batch_test	1689
5.342.2.2 test_data	1689

5.342.3 Enumeration Type Documentation	1689
5.342.3.1 batch_test_t	1689
5.342.4 Function Documentation	1689
5.342.4.1 main	1689
5.342.5 Variable Documentation	1689
5.342.5.1 batch_point_buffer	1689
5.342.5.2 curved25519_expected	1689
5.342.5.3 dataset	1689
5.343src/providers/dime/error_codes.c File Reference	1690
5.343.1 Variable Documentation	1690
5.343.1.1 ERR_BAD_PARAM	1690
5.343.1.2 ERR_CRYPT0	1690
5.343.1.3 ERR_ENCODING	1690
5.343.1.4 ERR_FILE_IO	1690
5.343.1.5 ERR_NOMEM	1690
5.344src/providers/dime/error_codes.h File Reference	1691
5.344.1 Enumeration Type Documentation	1691
5.344.1.1 dime_errcode_t	1691
5.344.2 Variable Documentation	1691
5.344.2.1 ERR_BAD_PARAM	1691
5.344.2.2 ERR_CRYPT0	1691
5.344.2.3 ERR_ENCODING	1691
5.344.2.4 ERR_FILE_IO	1692
5.344.2.5 ERR_NOMEM	1692
5.345src/providers/dime/sds/sds.c File Reference	1693
5.345.1 Define Documentation	1694
5.345.1.1 SDS_LLSTR_SIZE	1694
5.345.2 Function Documentation	1694
5.345.2.1 hex_digit_to_int	1694
5.345.2.2 is_hex_digit	1694
5.345.2.3 sdsAllocPtr	1694
5.345.2.4 sdsAllocSize	1694
5.345.2.5 sdscat	1694
5.345.2.6 sdscatfmt	1694
5.345.2.7 sdscatlen	1694
5.345.2.8 sdscatprintf	1695
5.345.2.9 sdscatrepr	1695
5.345.2.10sdscatsds	1695
5.345.2.11sdscatvprintf	1695

5.345.2.12	sds	clear	1695
5.345.2.13	sds	cmp	1695
5.345.2.14	sds	cpy	1695
5.345.2.15	sds	cpylen	1695
5.345.2.16	sds	dup	1696
5.345.2.17	sds	empty	1696
5.345.2.18	sds	free	1696
5.345.2.19	sds	freesplitres	1696
5.345.2.20	sds	fromlonglong	1696
5.345.2.21	sds	growzero	1696
5.345.2.22	sds	IncrLen	1696
5.345.2.23	sds	join	1696
5.345.2.24	sds	joinsds	1697
5.345.2.25	sds	ll2str	1697
5.345.2.26	sds	MakeRoomFor	1697
5.345.2.27	sds	mapchars	1697
5.345.2.28	sds	new	1697
5.345.2.29	sds	newlen	1697
5.345.2.30	sds	range	1697
5.345.2.31	sds	RemoveFreeSpace	1697
5.345.2.32	sds	splitargs	1697
5.345.2.33	sds	splitlen	1698
5.345.2.34	sds	stolower	1698
5.345.2.35	sds	stoupper	1698
5.345.2.36	sds	strim	1698
5.345.2.37	sds	ull2str	1698
5.345.2.38	sds	updatelen	1698
5.346	src/providers/dime/sds/sds.h	File Reference	1699
5.346.1	Define Documentation		1700
5.346.1.1	SDS_HDR		1700
5.346.1.2	SDS_HDR_VAR		1700
5.346.1.3	SDS_MAX_PREALLOC		1700
5.346.1.4	SDS_TYPE_16		1700
5.346.1.5	SDS_TYPE_32		1700
5.346.1.6	SDS_TYPE_5		1700
5.346.1.7	SDS_TYPE_5_LEN		1700
5.346.1.8	SDS_TYPE_64		1701
5.346.1.9	SDS_TYPE_8		1701
5.346.1.10	SDS_TYPE_BITS		1701

5.346.1.1	ISDS_TYPE_MASK	1701
5.346.2	Typedef Documentation	1701
5.346.2.1	sds	1701
5.346.3	Function Documentation	1701
5.346.3.1	__attribute__	1701
5.346.3.2	sdsAllocPtr	1701
5.346.3.3	sdsAllocSize	1701
5.346.3.4	sdscat	1701
5.346.3.5	sdscatfmt	1702
5.346.3.6	sdscatlen	1702
5.346.3.7	sdscatprintf	1702
5.346.3.8	sdscatrepr	1702
5.346.3.9	sdscatsds	1702
5.346.3.10	sdscatvprintf	1702
5.346.3.11	sdsclear	1702
5.346.3.12	sdscmp	1702
5.346.3.13	sdscopy	1702
5.346.3.14	sdscopylen	1703
5.346.3.15	sdsdup	1703
5.346.3.16	sdsempty	1703
5.346.3.17	sdsfree	1703
5.346.3.18	sdsfreesplitres	1703
5.346.3.19	sdsfromlonglong	1703
5.346.3.20	sdsgrowzero	1703
5.346.3.21	sdsIncrLen	1703
5.346.3.22	sdsjoin	1704
5.346.3.23	sdsjoinsds	1704
5.346.3.24	sdsMakeRoomFor	1704
5.346.3.25	sdsmapchars	1704
5.346.3.26	sdsnew	1704
5.346.3.27	sdsnewlen	1704
5.346.3.28	sdsrange	1704
5.346.3.29	sdsRemoveFreeSpace	1704
5.346.3.30	sdsplitargs	1704
5.346.3.31	sdsplitlen	1705
5.346.3.32	sdstolower	1705
5.346.3.33	sdstoupper	1705
5.346.3.34	sdstrim	1705
5.346.3.35	sdsupdatelen	1705

5.347src/providers/dime/sds/sdsalloc.h File Reference	1706
5.347.1 Define Documentation	1706
5.347.1.1 s_free	1706
5.347.1.2 s_malloc	1706
5.347.1.3 s_realloc	1706
5.348src/providers/dime/sds/testhelp.h File Reference	1707
5.348.1 Define Documentation	1707
5.348.1.1 test_cond	1707
5.348.1.2 test_report	1707
5.348.2 Variable Documentation	1707
5.348.2.1 __failed_tests	1707
5.348.2.2 __test_num	1707
5.349src/providers/dime/signet-resolver/cache_pub.c File Reference	1708
5.349.1 Function Documentation	1708
5.349.1.1 add_cached_object	1708
5.349.1.2 add_cached_object_cmp	1708
5.349.1.3 add_cached_object_cmp_forced	1708
5.349.1.4 add_cached_object_forced	1709
5.349.1.5 cached_object_exists	1709
5.349.1.6 cached_object_exists_cmp	1709
5.349.1.7 destroy_cache_entry	1709
5.349.1.8 find_cached_object	1709
5.349.1.9 find_cached_object_cmp	1709
5.349.1.10get_cache_location	1709
5.349.1.11get_cache_obj_data	1709
5.349.1.12get_dime_dir_location	1709
5.349.1.13load_cache_contents	1710
5.349.1.14remove_cached_object	1710
5.349.1.15remove_cached_object_cmp	1710
5.349.1.16save_cache_contents	1710
5.349.1.17set_cache_location	1710
5.349.1.18set_cache_permissions	1710
5.350src/providers/dime/signet-resolver/dmtp.c File Reference	1711
5.350.1 Function Documentation	1712
5.350.1.1 _dx_connect_dual	1712
5.350.1.2 _dx_connect_standard	1713
5.350.1.3 _get_signet	1713
5.350.1.4 _sgnt_resolv_destroy_dmtp_session	1714
5.350.1.5 _sgnt_resolv_dmtp_connect	1714

5.350.1.6 _sgnt_resolv_dmtplib_data	1714
5.350.1.7 _sgnt_resolv_dmtplib_ehlo	1714
5.350.1.8 _sgnt_resolv_dmtplib_expect_banner	1715
5.350.1.9 _sgnt_resolv_dmtplib_get_mode	1715
5.350.1.10 _sgnt_resolv_dmtplib_get_signet	1715
5.350.1.11 _sgnt_resolv_dmtplib_help	1716
5.350.1.12 _sgnt_resolv_dmtplib_history	1716
5.350.1.13 _sgnt_resolv_dmtplib_initiate_starttls	1716
5.350.1.14 _sgnt_resolv_dmtplib_issue_command	1717
5.350.1.15 _sgnt_resolv_dmtplib_mail_from	1717
5.350.1.16 _sgnt_resolv_dmtplib_noop	1717
5.350.1.17 _sgnt_resolv_dmtplib_quit	1717
5.350.1.18 _sgnt_resolv_dmtplib_rcpt_to	1718
5.350.1.19 _sgnt_resolv_dmtplib_reset	1718
5.350.1.20 _sgnt_resolv_dmtplib_send_and_read	1718
5.350.1.21 _sgnt_resolv_dmtplib_stats	1719
5.350.1.22 _sgnt_resolv_dmtplib_str_to_mode	1719
5.350.1.23 _sgnt_resolv_dmtplib_verify_signet	1719
5.350.1.24 _sgnt_resolv_dmtplib_write_data	1719
5.350.1.25 _sgnt_resolv_parse_line_code	1720
5.350.1.26 _sgnt_resolv_read_dmtplib_line	1720
5.350.1.27 _sgnt_resolv_read_dmtplib_multiline	1720
5.350.1.28 _verify_dx_certificate	1721
5.351 src/servers/dmtplib/dmtplib.c File Reference	1722
5.351.1 Function Documentation	1722
5.351.1.1 dmtplib_data	1722
5.351.1.2 dmtplib_ehlo	1723
5.351.1.3 dmtplib_helo	1723
5.351.1.4 dmtplib_help	1723
5.351.1.5 dmtplib_hist	1723
5.351.1.6 dmtplib_init	1723
5.351.1.7 dmtplib_invalid	1724
5.351.1.8 dmtplib_mail	1724
5.351.1.9 dmtplib_mode	1724
5.351.1.10 dmtplib_noop	1724
5.351.1.11 dmtplib_quit	1725
5.351.1.12 dmtplib_rcpt	1725
5.351.1.13 dmtplib_rset	1725
5.351.1.14 dmtplib_sgnt	1726

5.351.1.15	dmtp_verb	1726
5.351.1.16	dmtp_vrfy	1726
5.352	src/providers/dime/signet-resolver/dmtp_pub.c File Reference	1727
5.352.1	Function Documentation	1727
5.352.1.1	dx_connect_dual	1727
5.352.1.2	dx_connect_standard	1727
5.352.1.3	get_signet	1727
5.352.1.4	sgnt_resolv_destroy_dmtp_session	1727
5.352.1.5	sgnt_resolv_dmtp_connect	1728
5.352.1.6	sgnt_resolv_dmtp_data	1728
5.352.1.7	sgnt_resolv_dmtp_ehlo	1728
5.352.1.8	sgnt_resolv_dmtp_get_mode	1728
5.352.1.9	sgnt_resolv_dmtp_get_signet	1728
5.352.1.10	sgnt_resolv_dmtp_help	1728
5.352.1.11	sgnt_resolv_dmtp_history	1728
5.352.1.12	sgnt_resolv_dmtp_mail_from	1728
5.352.1.13	sgnt_resolv_dmtp_noop	1728
5.352.1.14	sgnt_resolv_dmtp_quit	1729
5.352.1.15	sgnt_resolv_dmtp_rcpt_to	1729
5.352.1.16	sgnt_resolv_dmtp_reset	1729
5.352.1.17	sgnt_resolv_dmtp_stats	1729
5.352.1.18	sgnt_resolv_dmtp_str_to_mode	1729
5.352.1.19	sgnt_resolv_dmtp_verify_signet	1729
5.352.1.20	verify_dx_certificate	1729
5.353	src/providers/dime/signet-resolver/dns.c File Reference	1730
5.353.1	Define Documentation	1732
5.353.1.1	INITIALIZE_DNS	1732
5.353.2	Function Documentation	1732
5.353.2.1	_add_dnskey_entry	1732
5.353.2.2	_add_dnskey_entry_rsa	1733
5.353.2.3	_add_ds_entry	1733
5.353.2.4	_clone_dnskey_record_cb	1734
5.353.2.5	_compare_rdata	1734
5.353.2.6	_compute_dnskey_sha_hash	1734
5.353.2.7	_deserialize_dnskey_record_cb	1735
5.353.2.8	_deserialize_ds_record_cb	1735
5.353.2.9	_destroy_dnskey	1735
5.353.2.10	_destroy_dnskey_record_cb	1736
5.353.2.11	_destroy_ds	1736

5.353.2.12_destroy_ds_record_cb	1736
5.353.2.13_dnskey_domain_comparator	1736
5.353.2.14_dnskey_tag_comparator	1737
5.353.2.15_ds_comparator	1737
5.353.2.16_dump_dns_header	1737
5.353.2.17_dump_dnskey_record_cb	1737
5.353.2.18_dump_ds_record_cb	1738
5.353.2.19_fixup_dnskey_validation	1738
5.353.2.20_fixup_ds_links	1738
5.353.2.21_free_mx_records	1739
5.353.2.22_get_dnskey_by_tag	1739
5.353.2.23_get_ds_by_dnskey	1739
5.353.2.24_get_keytag	1739
5.353.2.25_get_mx_records	1740
5.353.2.26_get_rsa_dnskey	1740
5.353.2.27_get_signing_key_names	1741
5.353.2.28_get_txt_record	1741
5.353.2.29_initialize_resolver	1741
5.353.2.30_is_validated_key	1741
5.353.2.31_load_dnskey_file	1742
5.353.2.32_lookup_dnskey	1742
5.353.2.33_lookup_ds	1742
5.353.2.34_mem_append_canon	1743
5.353.2.35_rsa_verify_record	1743
5.353.2.36_serialize_dnskey_record_cb	1744
5.353.2.37_serialize_ds_record_cb	1744
5.353.2.38_sort_rrs_canonical	1744
5.353.2.39_validate_rrsig_rr	1745
5.354src/providers/dime/signet-resolver/dns.h File Reference	1746
5.354.1 Define Documentation	1748
5.354.1.1 DNSKEY_RR_FLAG_SEP	1748
5.354.1.2 DNSKEY_RR_FLAG_ZK	1749
5.354.1.3 DNSKEY_RR_LEN	1749
5.354.1.4 DNSKEY_RR_PROTO	1749
5.354.1.5 DNSSEC_DIGEST_SHA1	1749
5.354.1.6 DNSSEC_DIGEST_SHA256	1749
5.354.1.7 IS_ROOT_LABEL	1749
5.354.1.8 NS_ALG_RSASHA1	1749
5.354.1.9 NS_ALG_RSASHA256	1749

5.354.1.10NS_ALG_RSASHA512	1749
5.354.1.11ROOT_KEY_FILE	1750
5.354.1.12T_DNSKEY	1750
5.354.1.13T_DS	1750
5.354.1.14T_RRSIG	1750
5.354.2 Typedef Documentation	1750
5.354.2.1 dnskey_t	1750
5.354.2.2 ds_t	1750
5.354.3 Function Documentation	1750
5.354.3.1 __attribute__	1750
5.354.3.2 _add_dnskey_entry	1751
5.354.3.3 _add_dnskey_entry_rsa	1751
5.354.3.4 _add_ds_entry	1752
5.354.3.5 _clone_dnskey_record_cb	1752
5.354.3.6 _compare_rdata	1752
5.354.3.7 _deserialize_dnskey_record_cb	1753
5.354.3.8 _deserialize_ds_record_cb	1753
5.354.3.9 _destroy_dnskey_record_cb	1753
5.354.3.10_destroy_ds_record_cb	1753
5.354.3.11_dnskey_domain_comparator	1754
5.354.3.12_dnskey_tag_comparator	1754
5.354.3.13_ds_comparator	1754
5.354.3.14_dump_dns_header	1754
5.354.3.15_dump_dnskey_record_cb	1755
5.354.3.16_dump_ds_record_cb	1755
5.354.3.17_fixup_dnskey_validation	1755
5.354.3.18_fixup_ds_links	1756
5.354.3.19_get_rsa_dnskey	1756
5.354.3.20_get_signing_key_names	1756
5.354.3.21_initialize_resolver	1757
5.354.3.22_mem_append_canon	1757
5.354.3.23_serialize_dnskey_record_cb	1757
5.354.3.24_serialize_ds_record_cb	1758
5.354.3.25_sort_rrs_canonical	1758
5.354.3.26PUBLIC_FUNC_DECL	1759
5.354.3.27PUBLIC_FUNC_DECL	1759
5.354.3.28PUBLIC_FUNC_DECL	1759
5.354.3.29PUBLIC_FUNC_DECL	1759
5.354.3.30PUBLIC_FUNC_DECL	1759

5.354.3.31PUBLIC_FUNC_DECL	1759
5.354.3.32PUBLIC_FUNC_DECL	1759
5.354.3.33PUBLIC_FUNC_DECL	1759
5.354.3.34PUBLIC_FUNC_DECL	1759
5.354.3.35PUBLIC_FUNC_DECL	1759
5.354.3.36PUBLIC_FUNC_DECL	1759
5.354.3.37PUBLIC_FUNC_DECL	1759
5.354.3.38PUBLIC_FUNC_DECL	1759
5.354.3.39PUBLIC_FUNC_DECL	1759
5.354.3.40PUBLIC_FUNC_DECL	1759
5.354.4 Variable Documentation	1759
5.354.4.1 dnskey_rr_flags_t	1759
5.354.4.2 ds_rr_t	1759
5.354.4.3 rrsig_rr_t	1759
5.355src/providers/dime/signet-resolver/mrec.c File Reference	1760
5.355.1 Function Documentation	1760
5.355.1.1 _deserialize_dime_record_cb	1760
5.355.1.2 _destroy_dime_record	1761
5.355.1.3 _destroy_dime_record_cb	1761
5.355.1.4 _dump_dime_record_cb	1761
5.355.1.5 _get_dime_record	1762
5.355.1.6 _get_dime_record_from_file	1762
5.355.1.7 _parse_dime_record	1762
5.355.1.8 _serialize_dime_record_cb	1763
5.355.1.9 _validate_dime_record	1763
5.356src/providers/dime/signet-resolver/mrec.h File Reference	1764
5.356.1 Define Documentation	1765
5.356.1.1 DIME_RECORD_DNS_PREFIX	1765
5.356.1.2 DIME_VERSION_NO	1765
5.356.2 Enumeration Type Documentation	1765
5.356.2.1 dime_msg_policy	1765
5.356.2.2 dime_sub_policy	1765
5.356.3 Function Documentation	1765
5.356.3.1 _deserialize_dime_record_cb	1765
5.356.3.2 _destroy_dime_record_cb	1766
5.356.3.3 _dump_dime_record_cb	1766
5.356.3.4 _serialize_dime_record_cb	1766
5.356.3.5 PUBLIC_FUNC_DECL	1767
5.356.3.6 PUBLIC_FUNC_DECL	1767

5.356.3.7 PUBLIC_FUNC_DECL	1767
5.356.3.8 PUBLIC_FUNC_DECL	1767
5.356.3.9 PUBLIC_FUNC_DECL	1767
5.357src/providers/dime/signet-resolver/mrec_pub.c File Reference	1768
5.357.1 Function Documentation	1768
5.357.1.1 destroy_dime_record	1768
5.357.1.2 get_dime_record	1768
5.357.1.3 get_dime_record_from_file	1768
5.357.1.4 parse_dime_record	1768
5.357.1.5 validate_dime_record	1768
5.358src/providers/dime/signet-resolver/signet-ssl.h File Reference	1769
5.358.1 Define Documentation	1770
5.358.1.1 CA_DIR	1770
5.358.1.2 CA_FILE	1770
5.358.1.3 CRL_FILE	1770
5.358.2 Function Documentation	1770
5.358.2.1 _deserialize_ocsp_response_cb	1770
5.358.2.2 _destroy_ocsp_response_cb	1771
5.358.2.3 _domain_wildcard_check	1771
5.358.2.4 _dump_ocsp_response_cb	1771
5.358.2.5 _get_cache_ocsp_id	1771
5.358.2.6 _get_cert_store	1772
5.358.2.7 _ocsp_response_callback	1772
5.358.2.8 _serialize_ocsp_response_cb	1773
5.358.2.9 _ssl_fd_loop	1773
5.358.2.10_validate_self_signed	1773
5.358.2.11_verify_certificate_callback	1773
5.358.2.12PUBLIC_FUNC_DECL	1774
5.358.2.13PUBLIC_FUNC_DECL	1774
5.358.2.14PUBLIC_FUNC_DECL	1774
5.358.2.15PUBLIC_FUNC_DECL	1774
5.358.2.16PUBLIC_FUNC_DECL	1774
5.358.2.17PUBLIC_FUNC_DECL	1774
5.358.2.18PUBLIC_FUNC_DECL	1774
5.358.2.19PUBLIC_FUNC_DECL	1774
5.358.2.20PUBLIC_FUNC_DECL	1774
5.358.2.21PUBLIC_FUNC_DECL	1774
5.359src/providers/dime/signet-resolver/ssl.c File Reference	1775
5.359.1 Function Documentation	1776

5.359.1.1 _deserialize_ocsp_response_cb	1776
5.359.1.2 _destroy_ocsp_response_cb	1777
5.359.1.3 _do_ocsp_validation	1777
5.359.1.4 _do_x509_hostname_check	1777
5.359.1.5 _do_x509_validation	1778
5.359.1.6 _domain_wildcard_check	1778
5.359.1.7 _dump_ocsp_response_cb	1778
5.359.1.8 _get_cache_ocsp_id	1779
5.359.1.9 _get_cert_store	1779
5.359.1.10 _get_cert_subject_cn	1779
5.359.1.11 _ocsp_response_callback	1780
5.359.1.12 _serialize_ocsp_response_cb	1780
5.359.1.13 _ssl_connect_host	1781
5.359.1.14 _ssl_disconnect	1781
5.359.1.15 _ssl_fd_loop	1781
5.359.1.16 _ssl_get_client_context	1781
5.359.1.17 _ssl_initialize	1782
5.359.1.18 _ssl_shutdown	1782
5.359.1.19 _ssl_starttls	1782
5.359.1.20 _validate_self_signed	1782
5.359.1.21 _verify_certificate_callback	1783
5.360src/providers/dime/signet-resolver/ssl_pub.c File Reference	1784
5.360.1 Function Documentation	1784
5.360.1.1 do_ocsp_validation	1784
5.360.1.2 do_x509_hostname_check	1784
5.360.1.3 do_x509_validation	1784
5.360.1.4 get_cert_subject_cn	1784
5.360.1.5 ssl_connect_host	1784
5.360.1.6 ssl_disconnect	1784
5.360.1.7 ssl_get_client_context	1785
5.360.1.8 ssl_initialize	1785
5.360.1.9 ssl_shutdown	1785
5.360.1.10 ssl_starttls	1785
5.361src/providers/dime/signet/general.c File Reference	1786
5.361.1 Define Documentation	1786
5.361.1.1 SKEY_EMPTY	1786
5.361.1.2 SKEY_SIZE1	1786
5.361.1.3 SKEY_SIZE2	1786
5.361.2 Function Documentation	1786

5.361.2.1	dime_number_to_str	1786
5.361.3	Variable Documentation	1787
5.361.3.1	signet_org_field_keys	1787
5.361.3.2	signet_ssr_field_keys	1787
5.361.3.3	signet_user_field_keys	1787
5.362	src/providers/dime/signet/signet.c File Reference	1788
5.362.1	Function Documentation	1791
5.362.1.1	dime_sgnt_enckey_fetch	1791
5.362.1.2	dime_sgnt_enckey_set	1791
5.362.1.3	dime_sgnt_fid_count_get	1791
5.362.1.4	dime_sgnt_fid_exists	1792
5.362.1.5	dime_sgnt_fid_num_fetch	1792
5.362.1.6	dime_sgnt_fid_num_remove	1792
5.362.1.7	dime_sgnt_field_defined_create	1793
5.362.1.8	dime_sgnt_field_defined_set	1793
5.362.1.9	dime_sgnt_field_undefined_create	1793
5.362.1.10	dime_sgnt_field_undefined_fetch	1794
5.362.1.11	dime_sgnt_field_undefined_remove	1794
5.362.1.12	dime_sgnt_file_create	1794
5.362.1.13	dime_sgnt_fingerprint_crypto	1795
5.362.1.14	dime_sgnt_fingerprint_full	1795
5.362.1.15	dime_sgnt_fingerprint_id	1795
5.362.1.16	dime_sgnt_fingerprint_ssr	1795
5.362.1.17	dime_sgnt_id_fetch	1796
5.362.1.18	dime_sgnt_id_set	1796
5.362.1.19	dime_sgnt_msg_sig_verify	1796
5.362.1.20	dime_sgnt_sig_coc_sign	1797
5.362.1.21	dime_sgnt_sig_crypto_sign	1797
5.362.1.22	dime_sgnt_sig_full_sign	1797
5.362.1.23	dime_sgnt_sig_id_sign	1798
5.362.1.24	dime_sgnt_sig_ssr_sign	1798
5.362.1.25	dime_sgnt_signet_b64_deserialize	1798
5.362.1.26	dime_sgnt_signet_b64_serialize	1798
5.362.1.27	dime_sgnt_signet_binary_deserialize	1799
5.362.1.28	dime_sgnt_signet_binary_serialize	1799
5.362.1.29	dime_sgnt_signet_create	1799
5.362.1.30	dime_sgnt_signet_create_w_keys	1800
5.362.1.31	dime_sgnt_signet_crypto_split	1800
5.362.1.32	dime_sgnt_signet_destroy	1800

5.362.1.33	dime_sgnt_signet_dump	1800
5.362.1.34	dime_sgnt_signet_dupe	1801
5.362.1.35	dime_sgnt_signet_full_split	1801
5.362.1.36	dime_sgnt_signet_load	1801
5.362.1.37	dime_sgnt_signkey_fetch	1801
5.362.1.38	dime_sgnt_signkey_set	1802
5.362.1.39	dime_sgnt_signkeys_msg_fetch	1802
5.362.1.40	dime_sgnt_signkeys_signet_fetch	1802
5.362.1.41	dime_sgnt_signkeys_software_fetch	1803
5.362.1.42	dime_sgnt_signkeys_tls_fetch	1803
5.362.1.43	dime_sgnt_sok_create	1803
5.362.1.44	dime_sgnt_sok_num_fetch	1804
5.362.1.45	dime_sgnt_state_to_str	1804
5.362.1.46	dime_sgnt_type_get	1804
5.362.1.47	dime_sgnt_type_set	1804
5.362.1.48	dime_sgnt_validate_all	1805
5.363	src/providers/dime/signet/signet.h File Reference	1806
5.363.1	Enumeration Type Documentation	1809
5.363.1.1	signet_state_t	1809
5.363.1.2	signet_type_t	1809
5.363.2	Function Documentation	1809
5.363.2.1	dime_sgnt_enckey_fetch	1809
5.363.2.2	dime_sgnt_enckey_set	1810
5.363.2.3	dime_sgnt_fid_count_get	1810
5.363.2.4	dime_sgnt_fid_exists	1810
5.363.2.5	dime_sgnt_fid_num_fetch	1811
5.363.2.6	dime_sgnt_fid_num_remove	1811
5.363.2.7	dime_sgnt_field_defined_create	1811
5.363.2.8	dime_sgnt_field_defined_set	1812
5.363.2.9	dime_sgnt_field_undefined_create	1812
5.363.2.10	dime_sgnt_field_undefined_fetch	1812
5.363.2.11	dime_sgnt_field_undefined_remove	1813
5.363.2.12	dime_sgnt_file_create	1813
5.363.2.13	dime_sgnt_fingerprint_crypto	1813
5.363.2.14	dime_sgnt_fingerprint_full	1814
5.363.2.15	dime_sgnt_fingerprint_id	1814
5.363.2.16	dime_sgnt_fingerprint_ssr	1814
5.363.2.17	dime_sgnt_id_fetch	1814
5.363.2.18	dime_sgnt_id_set	1815

5.363.2.19	dime_sgnt_msg_sig_verify	1815
5.363.2.20	dime_sgnt_sig_coc_sign	1815
5.363.2.21	dime_sgnt_sig_crypto_sign	1816
5.363.2.22	dime_sgnt_sig_full_sign	1816
5.363.2.23	dime_sgnt_sig_id_sign	1816
5.363.2.24	dime_sgnt_sig_ssr_sign	1816
5.363.2.25	dime_sgnt_signet_b64_deserialize	1817
5.363.2.26	dime_sgnt_signet_b64_serialize	1817
5.363.2.27	dime_sgnt_signet_binary_deserialize	1817
5.363.2.28	dime_sgnt_signet_binary_serialize	1818
5.363.2.29	dime_sgnt_signet_create	1818
5.363.2.30	dime_sgnt_signet_create_w_keys	1818
5.363.2.31	dime_sgnt_signet_crypto_split	1818
5.363.2.32	dime_sgnt_signet_destroy	1819
5.363.2.33	dime_sgnt_signet_dump	1819
5.363.2.34	dime_sgnt_signet_dupe	1819
5.363.2.35	dime_sgnt_signet_full_split	1819
5.363.2.36	dime_sgnt_signet_load	1820
5.363.2.37	dime_sgnt_signkey_fetch	1820
5.363.2.38	dime_sgnt_signkey_set	1820
5.363.2.39	dime_sgnt_signkeys_msg_fetch	1821
5.363.2.40	dime_sgnt_signkeys_signet_fetch	1821
5.363.2.41	dime_sgnt_signkeys_software_fetch	1821
5.363.2.42	dime_sgnt_signkeys_tls_fetch	1822
5.363.2.43	dime_sgnt_sok_create	1822
5.363.2.44	dime_sgnt_sok_num_fetch	1822
5.363.2.45	dime_sgnt_state_to_str	1823
5.363.2.46	dime_sgnt_type_get	1823
5.363.2.47	dime_sgnt_type_set	1823
5.363.2.48	dime_sgnt_validate_all	1823
5.364	src/providers/dime/util/encoding.c File Reference	1825
5.364.1	Function Documentation	1825
5.364.1.1	libdime_base64_decode	1825
5.364.1.2	libdime_base64_encode	1825
5.365	src/providers/dime/util/encoding.h File Reference	1826
5.365.1	Function Documentation	1826
5.365.1.1	libdime_base64_decode	1826
5.365.1.2	libdime_base64_encode	1826
5.366	src/providers/dime/util/encrypt.c File Reference	1827

5.366.1 Define Documentation	1827
5.366.1.1 LOG_OPENSSL_ERROR	1827
5.366.2 Function Documentation	1828
5.366.2.1 encrypt_ctx_free	1828
5.366.2.2 encrypt_ctx_new	1828
5.366.2.3 encrypt_deserialize_privkey	1828
5.366.2.4 encrypt_deserialize_pubkey	1828
5.366.2.5 encrypt_keypair_generate	1829
5.367src/providers/dime/util/encrypt.h File Reference	1830
5.367.1 Typedef Documentation	1830
5.367.1.1 encrypt_ctx_t	1830
5.367.2 Function Documentation	1830
5.367.2.1 encrypt_ctx_free	1830
5.367.2.2 encrypt_ctx_new	1831
5.367.2.3 encrypt_deserialize_privkey	1831
5.367.2.4 encrypt_deserialize_pubkey	1831
5.367.2.5 encrypt_keypair_generate	1831
5.368src/providers/images/freetype.c File Reference	1832
5.368.1 Function Documentation	1832
5.368.1.1 lib_load_freetype	1832
5.368.1.2 lib_version_freetype	1832
5.368.2 Variable Documentation	1832
5.368.2.1 lib_magma	1832
5.369src/providers/images/gd.c File Reference	1833
5.369.1 Function Documentation	1833
5.369.1.1 lib_load_gd	1833
5.369.1.2 lib_version_gd	1833
5.370src/providers/images/images.h File Reference	1834
5.370.1 Function Documentation	1834
5.370.1.1 lib_load_freetype	1834
5.370.1.2 lib_load_gd	1834
5.370.1.3 lib_load_jpeg	1835
5.370.1.4 lib_load_png	1835
5.370.1.5 lib_version_freetype	1835
5.370.1.6 lib_version_gd	1835
5.370.1.7 lib_version_jpeg	1835
5.370.1.8 lib_version_png	1836
5.371src/providers/images/jpeg.c File Reference	1837
5.371.1 Function Documentation	1837

5.371.1.1 lib_load_jpeg	1837
5.371.1.2 lib_version_jpeg	1837
5.372src/providers/images/png.c File Reference	1838
5.372.1 Function Documentation	1838
5.372.1.1 lib_load_png	1838
5.372.1.2 lib_version_png	1838
5.373src/providers/parsers/json.c File Reference	1839
5.373.1 Function Documentation	1839
5.373.1.1 lib_load_jansson	1839
5.373.1.2 lib_version_jansson	1839
5.374src/providers/parsers/utf8.c File Reference	1840
5.374.1 Function Documentation	1840
5.374.1.1 lib_load_utf8proc	1840
5.374.1.2 lib_version_utf8proc	1840
5.374.1.3 utf8_error_string	1841
5.374.1.4 utf8_length_st	1841
5.374.1.5 utf8_valid_st	1841
5.375src/providers/parsers/xml.c File Reference	1842
5.375.1 Function Documentation	1844
5.375.1.1 lib_load_xml	1844
5.375.1.2 lib_version_xml	1844
5.375.1.3 xml_create_doc	1844
5.375.1.4 xml_create_parser_ctx	1845
5.375.1.5 xml_create_xpath_ctx	1845
5.375.1.6 xml_dump_doc	1845
5.375.1.7 xml_encode	1846
5.375.1.8 xml_error	1846
5.375.1.9 xml_free_doc	1846
5.375.1.10xml_free_parser_ctx	1847
5.375.1.11xml_free_xpath_ctx	1847
5.375.1.12xml_free_xpath_obj	1847
5.375.1.13xml_get_xpath_int16	1847
5.375.1.14xml_get_xpath_int32	1848
5.375.1.15xml_get_xpath_int64	1848
5.375.1.16xml_get_xpath_int8	1848
5.375.1.17xml_get_xpath_node_count	1849
5.375.1.18xml_get_xpath_ns	1849
5.375.1.19xml_get_xpath_st	1849
5.375.1.20xml_get_xpath_uint16	1849

5.375.1.21xml_get_xpath_uint32	1850
5.375.1.22xml_get_xpath_uint64	1850
5.375.1.23xml_get_xpath_uint8	1850
5.375.1.24xml_node_add_sibling	1850
5.375.1.25xml_node_free	1851
5.375.1.26xml_node_get_content_st	1851
5.375.1.27xml_node_new	1851
5.375.1.28xml_node_set_content	1852
5.375.1.29xml_node_set_property	1852
5.375.1.30xml_set_xpath_ns	1852
5.375.1.31xml_set_xpath_property	1853
5.375.1.32xml_set_xpath_uint64	1853
5.375.1.33xml_start	1853
5.375.1.34xml_stop	1853
5.375.1.35xml_xpath_eval	1854
5.375.1.36xml_xpath_set_namespace	1854
5.375.2 Variable Documentation	1854
5.375.2.1 log_mutex	1854
5.375.2.2 xml_version_string	1854
5.376src/providers/prime/cryptography/aes.c File Reference	1855
5.376.1 Define Documentation	1855
5.376.1.1 PRIME_CHUNK_HEAD_LEN	1855
5.376.1.2 PRIME_CHUNK_KEY_LEN	1855
5.376.1.3 PRIME_CHUNK_PREFIX_LEN	1856
5.376.1.4 PRIME_OBJECT_HEAD_LEN	1856
5.376.1.5 PRIME_OBJECT_KEY_LEN	1856
5.376.1.6 PRIME_OBJECT_PAYLOAD_PREFIX_LEN	1856
5.376.1.7 PRIME_OBJECT_PREFIX_LEN	1856
5.376.2 Function Documentation	1856
5.376.2.1 __attribute__	1856
5.376.2.2 aes_artifact_decrypt	1856
5.376.2.3 aes_artifact_encrypt	1856
5.376.2.4 aes_chunk_decrypt	1857
5.376.2.5 aes_chunk_encrypt	1857
5.376.2.6 aes_cipher_key	1857
5.376.2.7 aes_tag_shard	1858
5.376.2.8 aes_vector_shard	1858
5.376.3 Variable Documentation	1858
5.376.3.1 prime_encrypted_chunk_header_t	1858

5.376.3.2 prime_encrypted_object_header_t	1858
5.376.3.3 prime_encrypted_payload_prefix_t	1858
5.377src/providers/prime/cryptography/secp256k1.c File Reference	1859
5.377.1 Function Documentation	1859
5.377.1.1 secp256k1_alloc	1859
5.377.1.2 secp256k1_compute_kek	1860
5.377.1.3 secp256k1_free	1860
5.377.1.4 secp256k1_generate	1860
5.377.1.5 secp256k1_private_get	1860
5.377.1.6 secp256k1_private_set	1861
5.377.1.7 secp256k1_public_get	1861
5.377.1.8 secp256k1_public_set	1861
5.377.1.9 secp256k1_type	1862
5.377.2 Variable Documentation	1862
5.377.2.1 prime_curve_group	1862
5.378src/providers/prime/keys/orgs.c File Reference	1863
5.378.1 Function Documentation	1863
5.378.1.1 org_encrypted_key_get	1863
5.378.1.2 org_encrypted_key_set	1863
5.378.1.3 org_key_alloc	1863
5.378.1.4 org_key_free	1863
5.378.1.5 org_key_generate	1864
5.378.1.6 org_key_get	1864
5.378.1.7 org_key_length	1864
5.378.1.8 org_key_set	1864
5.379src/providers/prime/signets/orgs.c File Reference	1865
5.379.1 Function Documentation	1865
5.379.1.1 org_signet_alloc	1865
5.379.1.2 org_signet_fingerprint	1865
5.379.1.3 org_signet_free	1865
5.379.1.4 org_signet_generate	1865
5.379.1.5 org_signet_get	1866
5.379.1.6 org_signet_length	1866
5.379.1.7 org_signet_set	1866
5.379.1.8 org_signet_verify	1866
5.380src/providers/prime/keys/users.c File Reference	1867
5.380.1 Function Documentation	1867
5.380.1.1 user_encrypted_key_get	1867
5.380.1.2 user_encrypted_key_set	1867

5.380.1.3 user_key_alloc	1867
5.380.1.4 user_key_free	1867
5.380.1.5 user_key_generate	1868
5.380.1.6 user_key_get	1868
5.380.1.7 user_key_length	1868
5.380.1.8 user_key_set	1868
5.381src/providers/prime/signets/users.c File Reference	1869
5.381.1 Function Documentation	1869
5.381.1.1 user_signet_alloc	1869
5.381.1.2 user_signet_fingerprint	1869
5.381.1.3 user_signet_free	1869
5.381.1.4 user_signet_get	1869
5.381.1.5 user_signet_length	1870
5.381.1.6 user_signet_set	1870
5.381.1.7 user_signet_verify_chain_of_custody	1870
5.381.1.8 user_signet_verify_org	1870
5.381.1.9 user_signet_verify_self	1870
5.382src/providers/prime/messages/chunks/chunks.c File Reference	1871
5.382.1 Function Documentation	1871
5.382.1.1 chunk_buffer_read	1871
5.382.1.2 chunk_buffer_size	1871
5.382.1.3 chunk_header_read	1871
5.382.1.4 chunk_header_size	1871
5.382.1.5 chunk_header_type	1872
5.382.1.6 chunk_header_write	1872
5.383src/providers/prime/messages/chunks/chunks.h File Reference	1873
5.383.1 Function Documentation	1874
5.383.1.1 chunk_buffer_read	1874
5.383.1.2 chunk_buffer_size	1874
5.383.1.3 chunk_header_read	1875
5.383.1.4 chunk_header_size	1875
5.383.1.5 chunk_header_type	1875
5.383.1.6 chunk_header_write	1875
5.383.1.7 encrypted_chunk_alloc	1875
5.383.1.8 encrypted_chunk_buffer	1875
5.383.1.9 encrypted_chunk_cleanup	1876
5.383.1.10encrypted_chunk_free	1876
5.383.1.11encrypted_chunk_get	1876
5.383.1.12encrypted_chunk_set	1876

5.383.1.13ephemeral_chunk_alloc	1876
5.383.1.14ephemeral_chunk_buffer	1876
5.383.1.15ephemeral_chunk_cleanup	1877
5.383.1.16ephemeral_chunk_free	1877
5.383.1.17ephemeral_chunk_get	1877
5.383.1.18ephemeral_chunk_set	1877
5.383.1.19eks_alloc	1877
5.383.1.20eks_cleanup	1877
5.383.1.21eks_free	1878
5.383.1.22eks_get	1878
5.383.1.23eks_set	1878
5.383.1.24signature_full_get	1878
5.383.1.25signature_full_verify	1878
5.383.1.26signature_tree_add	1879
5.383.1.27signature_tree_alloc	1879
5.383.1.28signature_tree_cleanup	1879
5.383.1.29signature_tree_free	1879
5.383.1.30signature_tree_get	1879
5.383.1.31signature_tree_verify	1879
5.383.1.32slots_actors	1880
5.383.1.33slots_alloc	1880
5.383.1.34slots_buffer	1880
5.383.1.35slots_cleanup	1880
5.383.1.36slots_count	1880
5.383.1.37slots_free	1880
5.383.1.38slots_get	1881
5.383.1.39slots_key	1881
5.383.1.40slots_set	1881
5.384src/providers/prime/messages/chunks/encrypted.c File Reference	1882
5.384.1 Function Documentation	1882
5.384.1.1 encrypted_chunk_alloc	1882
5.384.1.2 encrypted_chunk_buffer	1882
5.384.1.3 encrypted_chunk_cleanup	1882
5.384.1.4 encrypted_chunk_free	1882
5.384.1.5 encrypted_chunk_get	1883
5.384.1.6 encrypted_chunk_set	1883
5.385src/providers/prime/messages/chunks/ephemeral.c File Reference	1884
5.385.1 Function Documentation	1884
5.385.1.1 ephemeral_chunk_alloc	1884

5.385.1.2 ephemeral_chunk_buffer	1884
5.385.1.3 ephemeral_chunk_cleanup	1884
5.385.1.4 ephemeral_chunk_free	1884
5.385.1.5 ephemeral_chunk_get	1885
5.385.1.6 ephemeral_chunk_set	1885
5.386src/providers/prime/messages/chunks/keks.c File Reference	1886
5.386.1 Function Documentation	1886
5.386.1.1 keks_alloc	1886
5.386.1.2 keks_cleanup	1886
5.386.1.3 keks_free	1886
5.386.1.4 keks_get	1886
5.386.1.5 keks_set	1887
5.387src/providers/prime/messages/chunks/signature.c File Reference	1888
5.387.1 Function Documentation	1888
5.387.1.1 signature_full_get	1888
5.387.1.2 signature_full_verify	1888
5.387.1.3 signature_tree_add	1889
5.387.1.4 signature_tree_alloc	1889
5.387.1.5 signature_tree_cleanup	1889
5.387.1.6 signature_tree_free	1889
5.387.1.7 signature_tree_get	1889
5.387.1.8 signature_tree_verify	1889
5.388src/providers/prime/messages/chunks/slots.c File Reference	1890
5.388.1 Define Documentation	1890
5.388.1.1 PRIME_ACTOR_AUTHOR	1890
5.388.1.2 PRIME_ACTOR_DESTINATION	1890
5.388.1.3 PRIME_ACTOR_NONE	1890
5.388.1.4 PRIME_ACTOR_ORIGIN	1891
5.388.1.5 PRIME_ACTOR_RECIPIENT	1891
5.388.2 Function Documentation	1891
5.388.2.1 slots_actors	1891
5.388.2.2 slots_alloc	1891
5.388.2.3 slots_buffer	1891
5.388.2.4 slots_cleanup	1891
5.388.2.5 slots_count	1891
5.388.2.6 slots_free	1892
5.388.2.7 slots_get	1892
5.388.2.8 slots_key	1892
5.388.2.9 slots_set	1892

5.389src/providers/prime/messages/parts/parts.c File Reference	1893
5.389.1 Detailed Description	1893
5.389.2 Function Documentation	1893
5.389.2.1 part_buffer	1893
5.389.2.2 part_decrypt	1893
5.389.2.3 part_encrypt	1893
5.390src/providers/prime/messages/parts/parts.h File Reference	1894
5.390.1 Detailed Description	1894
5.390.2 Function Documentation	1894
5.390.2.1 part_buffer	1894
5.390.2.2 part_decrypt	1894
5.390.2.3 part_encrypt	1894
5.391src/providers/prime/prime.c File Reference	1895
5.391.1 Function Documentation	1896
5.391.1.1 prime_alloc	1896
5.391.1.2 prime_cleanup	1896
5.391.1.3 prime_free	1896
5.391.1.4 prime_get	1896
5.391.1.5 prime_key_decrypt	1896
5.391.1.6 prime_key_encrypt	1897
5.391.1.7 prime_key_generate	1897
5.391.1.8 prime_message_decrypt	1897
5.391.1.9 prime_message_encrypt	1897
5.391.1.10prime_request_generate	1897
5.391.1.11prime_request_sign	1898
5.391.1.12prime_set	1898
5.391.1.13prime_signet_fingerprint	1898
5.391.1.14prime_signet_generate	1898
5.391.1.15prime_signet_validate	1898
5.391.1.16prime_start	1898
5.391.1.17prime_stop	1899
5.391.2 Variable Documentation	1899
5.391.2.1 org_key	1899
5.391.2.2 org_signet	1899
5.391.2.3 prime_curve_group	1899
5.392src/providers/prime/prime.h File Reference	1900
5.392.1 Define Documentation	1902
5.392.1.1 AES_BLOCK_LEN	1902
5.392.1.2 AES_KEY_LEN	1902

5.392.1.3 AES_TAG_LEN	1902
5.392.1.4 AES_VECTOR_LEN	1903
5.392.1.5 ED25519_KEY_PRIV_LEN	1903
5.392.1.6 ED25519_KEY_PUB_LEN	1903
5.392.1.7 ED25519_SIGNATURE_LEN	1903
5.392.1.8 SECP256K1_KEY_PRIV_LEN	1903
5.392.1.9 SECP256K1_KEY_PUB_LEN	1903
5.392.1.10SECP256K1_SHARED_SECRET_LEN	1903
5.392.2 Typedef Documentation	1904
5.392.2.1 prime_type_t	1904
5.392.2.2 secp256k1_key_t	1904
5.392.3 Enumeration Type Documentation	1904
5.392.3.1 ed25519_key_type_t	1904
5.392.3.2 prime_artifact_type_t	1904
5.392.3.3 prime_encoding_t	1904
5.392.3.4 prime_flags_t	1904
5.392.3.5 prime_message_chunk_flags_t	1905
5.392.3.6 prime_message_chunk_type_t	1905
5.392.3.7 prime_message_type_t	1905
5.392.3.8 prime_org_artifact_fields_t	1906
5.392.3.9 prime_user_artifact_fields_t	1906
5.392.3.10secp256k1_key_type_t	1906
5.392.4 Function Documentation	1906
5.392.4.1 __attribute__	1906
5.392.4.2 prime_alloc	1907
5.392.4.3 prime_cleanup	1907
5.392.4.4 prime_free	1907
5.392.4.5 prime_get	1907
5.392.4.6 prime_key_decrypt	1908
5.392.4.7 prime_key_encrypt	1908
5.392.4.8 prime_key_generate	1908
5.392.4.9 prime_message_decrypt	1908
5.392.4.10prime_message_encrypt	1908
5.392.4.11prime_request_generate	1908
5.392.4.12prime_request_sign	1909
5.392.4.13prime_set	1909
5.392.4.14prime_signet_fingerprint	1909
5.392.4.15prime_signet_generate	1909
5.392.4.16prime_signet_validate	1909

5.392.4.17	prime_start	1909
5.392.4.18	prime_stop	1910
5.392.5	Variable Documentation	1910
5.392.5.1	ed25519_key_t	1910
5.392.5.2	org_key	1910
5.392.5.3	org_signet	1910
5.392.5.4	prime_chunk_keks_t	1910
5.392.5.5	prime_chunk_keys_t	1910
5.392.5.6	prime_chunk_slots_t	1911
5.392.5.7	prime_encrypted_chunk_t	1911
5.392.5.8	prime_ephemeral_chunk_t	1911
5.392.5.9	prime_message_t	1911
5.392.5.10	prime_org_key_t	1911
5.392.5.11	prime_org_signet_t	1911
5.392.5.12	prime_signature_tree_t	1911
5.392.5.13	prime_t	1911
5.392.5.14	prime_user_key_t	1911
5.392.5.15	prime_user_signet_t	1912
5.393	src/providers/prime/signets/requests.c File Reference	1913
5.393.1	Function Documentation	1913
5.393.1.1	user_request_generate	1913
5.393.1.2	user_request_get	1913
5.393.1.3	user_request_length	1913
5.393.1.4	user_request_rotation	1913
5.393.1.5	user_request_set	1914
5.393.1.6	user_request_sign	1914
5.393.1.7	user_request_verify_chain_of_custody	1914
5.393.1.8	user_request_verify_self	1914
5.394	src/providers/prime/signets/signets.h File Reference	1915
5.394.1	Function Documentation	1915
5.394.1.1	org_signet_alloc	1915
5.394.1.2	org_signet_fingerprint	1916
5.394.1.3	org_signet_free	1916
5.394.1.4	org_signet_generate	1916
5.394.1.5	org_signet_get	1916
5.394.1.6	org_signet_length	1916
5.394.1.7	org_signet_set	1916
5.394.1.8	org_signet_verify	1916
5.394.1.9	user_request_generate	1917

5.394.1.10	<code>user_request_get</code>	1917
5.394.1.11	<code>user_request_length</code>	1917
5.394.1.12	<code>user_request_rotation</code>	1917
5.394.1.13	<code>user_request_set</code>	1917
5.394.1.14	<code>user_request_sign</code>	1917
5.394.1.15	<code>user_request_verify_chain_of_custody</code>	1918
5.394.1.16	<code>user_request_verify_self</code>	1918
5.394.1.17	<code>user_signet_alloc</code>	1918
5.394.1.18	<code>user_signet_fingerprint</code>	1918
5.394.1.19	<code>user_signet_free</code>	1918
5.394.1.20	<code>user_signet_get</code>	1918
5.394.1.21	<code>user_signet_length</code>	1919
5.394.1.22	<code>user_signet_set</code>	1919
5.394.1.23	<code>user_signet_verify_chain_of_custody</code>	1919
5.394.1.24	<code>user_signet_verify_org</code>	1919
5.394.1.25	<code>user_signet_verify_self</code>	1919
5.395	<code>src/providers/prime/transposition/armored/armored.h</code> File Reference	1920
5.395.1	Function Documentation	1920
5.395.1.1	<code>prime_pem_begin</code>	1920
5.395.1.2	<code>prime_pem_end</code>	1920
5.395.1.3	<code>prime_pem_unwrap</code>	1920
5.395.1.4	<code>prime_pem_wrap</code>	1920
5.396	<code>src/providers/prime/transposition/armored/pem.c</code> File Reference	1921
5.396.1	Define Documentation	1921
5.396.1.1	<code>PRIME_PEM_LINE_WRAP_CHARS</code>	1921
5.396.1.2	<code>PRIME_PEM_LINE_WRAP_LENGTH</code>	1921
5.396.1.3	<code>PRIME_PEM_LINE_WRAP_TYPE</code>	1921
5.396.2	Function Documentation	1921
5.396.2.1	<code>prime_pem_begin</code>	1921
5.396.2.2	<code>prime_pem_end</code>	1922
5.396.2.3	<code>prime_pem_unwrap</code>	1922
5.396.2.4	<code>prime_pem_wrap</code>	1922
5.396.3	Variable Documentation	1922
5.396.3.1	<code>prime_pem_endings</code>	1922
5.396.3.2	<code>prime_pem_headings</code>	1923
5.397	<code>src/providers/prime/transposition/binary/binary.h</code> File Reference	1924
5.397.1	Typedef Documentation	1925
5.397.1.1	<code>prime_field_data_t</code>	1925
5.397.1.2	<code>prime_field_type_t</code>	1925

5.397.1.3	prime_size_t	1925
5.397.2	Function Documentation	1925
5.397.2.1	__attribute__	1925
5.397.2.2	prime_field_get	1926
5.397.2.3	prime_field_size_length	1926
5.397.2.4	prime_field_size_max	1926
5.397.2.5	prime_field_write	1926
5.397.2.6	prime_header_encrypted_message_write	1926
5.397.2.7	prime_header_encrypted_org_key_write	1926
5.397.2.8	prime_header_encrypted_user_key_write	1927
5.397.2.9	prime_header_length	1927
5.397.2.10	prime_header_org_key_write	1927
5.397.2.11	prime_header_org_signet_write	1927
5.397.2.12	prime_header_read	1927
5.397.2.13	prime_header_user_key_write	1928
5.397.2.14	prime_header_user_signet_write	1928
5.397.2.15	prime_header_user_signing_request_write	1928
5.397.2.16	prime_header_write	1928
5.397.2.17	prime_object_alloc	1928
5.397.2.18	prime_object_free	1929
5.397.2.19	prime_object_size_max	1929
5.397.2.20	prime_object_size_min	1929
5.397.2.21	prime_object_type	1929
5.397.2.22	prime_reader_open	1929
5.397.2.23	prime_reader_payload	1929
5.397.2.24	prime_reader_size	1930
5.397.2.25	prime_reader_type	1930
5.397.2.26	prime_unpack	1930
5.397.2.27	prime_unpack_fields	1930
5.397.2.28	prime_unpack_validate	1931
5.397.3	Variable Documentation	1931
5.397.3.1	prime_field_t	1931
5.397.3.2	prime_object_t	1931
5.397.3.3	prime_reader_t	1931
5.398	src/providers/prime/transposition/binary/fields.c File Reference	1932
5.398.1	Function Documentation	1932
5.398.1.1	prime_field_get	1932
5.398.1.2	prime_field_size_length	1932
5.398.1.3	prime_field_size_max	1932

5.398.1.4	prime_field_write	1932
5.399	src/providers/prime/transposition/binary/reader.c File Reference	1934
5.399.1	Function Documentation	1934
5.399.1.1	prime_reader_open	1934
5.399.1.2	prime_reader_payload	1934
5.399.1.3	prime_reader_size	1934
5.399.1.4	prime_reader_type	1935
5.400	src/providers/prime/transposition/binary/unpack.c File Reference	1936
5.400.1	Function Documentation	1936
5.400.1.1	prime_unpack	1936
5.400.1.2	prime_unpack_fields	1936
5.400.1.3	prime_unpack_validate	1936
5.401	src/providers/prime/transposition/transposition.h File Reference	1937
5.401.1	Define Documentation	1937
5.401.1.1	PRIME_FIXED_SIZE	1937
5.401.1.2	PRIME_MAX_1_BYTE	1937
5.401.1.3	PRIME_MAX_2_BYTE	1937
5.401.1.4	PRIME_MAX_3_BYTE	1937
5.401.1.5	PRIME_MAX_4_BYTE	1937
5.402	src/providers/providers.h File Reference	1938
5.402.1	Function Documentation	1938
5.402.1.1	__attribute__	1938
5.402.1.2	lib_load	1938
5.402.1.3	lib_symbols	1939
5.402.1.4	lib_unload	1939
5.402.2	Variable Documentation	1939
5.402.2.1	symbol_t	1939
5.403	src/providers/stacie/creation.c File Reference	1940
5.403.1	Detailed Description	1940
5.403.2	Function Documentation	1940
5.403.2.1	stacie_create_nonce	1940
5.403.2.2	stacie_create_salt	1940
5.403.2.3	stacie_create_shard	1941
5.404	src/providers/stacie/passwords.c File Reference	1942
5.404.1	Detailed Description	1942
5.404.2	Function Documentation	1942
5.404.2.1	stacie_derive_key	1942
5.404.2.2	stacie_derive_rounds	1943
5.404.2.3	stacie_derive_seed	1943

5.405src/providers/stacie/realms.c File Reference	1945
5.405.1 Detailed Description	1945
5.405.2 Function Documentation	1945
5.405.2.1 stacie_realm_cipher	1945
5.405.2.2 stacie_realm_key	1945
5.405.2.3 stacie_realm_tag	1946
5.405.2.4 stacie_realm_vector	1946
5.406src/providers/stacie/stacie.h File Reference	1947
5.406.1 Detailed Description	1948
5.406.2 Define Documentation	1948
5.406.2.1 STACIE_BLOCK_LENGTH	1948
5.406.2.2 STACIE_ENCRYPT_MAX	1948
5.406.2.3 STACIE_ENCRYPT_MIN	1948
5.406.2.4 STACIE_ENVELOPE_LENGTH	1948
5.406.2.5 STACIE_KEY_LENGTH	1948
5.406.2.6 STACIE_KEY_ROUNDS_MAX	1949
5.406.2.7 STACIE_KEY_ROUNDS_MIN	1949
5.406.2.8 STACIE_NONCE_LENGTH	1949
5.406.2.9 STACIE_SALT_LENGTH	1949
5.406.2.10STACIE_SHARD_LENGTH	1949
5.406.2.11STACIE_TOKEN_LENGTH	1949
5.406.2.12STACIE_TOKEN_ROUNDS	1949
5.406.3 Function Documentation	1949
5.406.3.1 stacie_create_nonce	1949
5.406.3.2 stacie_create_salt	1950
5.406.3.3 stacie_create_shard	1950
5.406.3.4 stacie_decrypt	1950
5.406.3.5 stacie_derive_key	1951
5.406.3.6 stacie_derive_rounds	1952
5.406.3.7 stacie_derive_seed	1952
5.406.3.8 stacie_derive_token	1953
5.406.3.9 stacie_encrypt	1953
5.406.3.10stacie_realm_cipher	1954
5.406.3.11stacie_realm_key	1954
5.406.3.12stacie_realm_tag	1955
5.406.3.13stacie_realm_vector	1955
5.407src/providers/stacie/tokens.c File Reference	1956
5.407.1 Detailed Description	1956
5.407.2 Function Documentation	1956

5.407.2.1 stacie_derive_token	1956
5.408src/providers/storage/storage.h File Reference	1958
5.408.1 Define Documentation	1959
5.408.1.1 TANK_ENTRY_VERSION	1959
5.408.1.2 TANK_RECORD_VERSION	1959
5.408.2 Enumeration Type Documentation	1959
5.408.2.1 "@79	1959
5.408.3 Function Documentation	1959
5.408.3.1 __attribute__	1959
5.408.3.2 lib_load_tokyo	1960
5.408.3.3 lib_version_tokyo	1960
5.408.3.4 tank_count	1960
5.408.3.5 tank_cycle	1960
5.408.3.6 tank_delete	1961
5.408.3.7 tank_delete_object	1961
5.408.3.8 tank_insert_object	1962
5.408.3.9 tank_load	1962
5.408.3.10 tank_maintain	1962
5.408.3.11 tank_size	1963
5.408.3.12 tank_start	1963
5.408.3.13 tank_stop	1963
5.408.3.14 tank_store	1963
5.408.3.15 tree_alloc	1964
5.408.3.16 tree_count	1964
5.408.4 Variable Documentation	1964
5.408.4.1 entry_t	1964
5.408.4.2 record_t	1965
5.408.4.3 TANK_FLAGS_E	1965
5.409src/providers/storage/tank.c File Reference	1966
5.409.1 Function Documentation	1966
5.409.1.1 tank_close	1966
5.409.1.2 tank_count	1967
5.409.1.3 tank_cycle	1967
5.409.1.4 tank_delete	1967
5.409.1.5 tank_load	1968
5.409.1.6 tank_maintain	1968
5.409.1.7 tank_open	1968
5.409.1.8 tank_size	1968
5.409.1.9 tank_start	1969

5.409.1.10	tank_stop	1969
5.409.1.11	tank_store	1969
5.409.2	Variable Documentation	1969
5.409.2.1	store	1969
5.409.2.2	system	1970
5.409.2.3	tanks	1970
5.409.2.4	tanks_lock	1970
5.409.2.5	tanks_next	1970
5.409.2.6	tanks_num	1970
5.409.2.7	tuner	1970
5.410	src/providers/storage/tokyo.c File Reference	1971
5.410.1	Function Documentation	1971
5.410.1.1	lib_load_tokyo	1971
5.410.1.2	lib_version_tokyo	1971
5.411	src/providers/storage/tree.c File Reference	1972
5.411.1	Function Documentation	1972
5.411.1.1	__attribute__	1972
5.411.1.2	tree_alloc	1972
5.411.1.3	tree_cmp	1973
5.411.1.4	tree_count	1973
5.411.1.5	tree_cursor_alloc	1973
5.411.1.6	tree_cursor_free	1974
5.411.1.7	tree_cursor_key_active	1974
5.411.1.8	tree_cursor_key_next	1974
5.411.1.9	tree_cursor_next	1974
5.411.1.10	tree_cursor_reset	1974
5.411.1.11	tree_cursor_value_active	1974
5.411.1.12	tree_cursor_value_next	1974
5.411.1.13	tree_delete	1974
5.411.1.14	tree_find	1975
5.411.1.15	tree_free	1975
5.411.1.16	tree_insert	1975
5.411.1.17	tree_truncate	1975
5.411.2	Variable Documentation	1976
5.411.2.1	tree_cursor_t	1976
5.412	src/providers/symbols.c File Reference	1977
5.412.1	Function Documentation	1984
5.412.1.1	lib_load	1984
5.412.1.2	lib_unload	1984

5.412.1.3	STACK_OF	1984
5.412.2	Variable Documentation	1984
5.412.2.1	__lzo_init_v2_d	1984
5.412.2.2	ASN1_STRING_data_d	1984
5.412.2.3	ASN1_STRING_TABLE_cleanup_d	1985
5.412.2.4	BIO_free_all_d	1985
5.412.2.5	BIO_free_d	1985
5.412.2.6	BIO_new_fp_d	1985
5.412.2.7	BIO_new_socket_d	1985
5.412.2.8	BIO_sock_cleanup_d	1985
5.412.2.9	BN_bn2bin_d	1985
5.412.2.10	BN_bn2dec_d	1985
5.412.2.11	BN_bn2hex_d	1985
5.412.2.12	BN_bn2mpi_d	1985
5.412.2.13	BN_cmp_d	1985
5.412.2.14	BN_CTX_free_d	1985
5.412.2.15	BN_CTX_new_d	1985
5.412.2.16	BN_CTX_start_d	1985
5.412.2.17	BN_free_d	1986
5.412.2.18	BN_hex2bn_d	1986
5.412.2.19	BN_mpi2bn_d	1986
5.412.2.20	BN_num_bits_d	1986
5.412.2.21	BUF_strlcat_d	1986
5.412.2.22	BZ2_bzBuffToBuffCompress_d	1986
5.412.2.23	BZ2_bzBuffToBuffDecompress_d	1986
5.412.2.24	BZ2_bzlibVersion_d	1986
5.412.2.25	cl_countsigs_d	1986
5.412.2.26	cl_engine_compile_d	1986
5.412.2.27	cl_engine_free_d	1986
5.412.2.28	cl_engine_new_d	1986
5.412.2.29	cl_engine_set_num_d	1987
5.412.2.30	cl_engine_set_str_d	1987
5.412.2.31	cl_init_d	1987
5.412.2.32	cl_load_d	1987
5.412.2.33	cl_retver_d	1987
5.412.2.34	cl_scandesc_d	1987
5.412.2.35	cl_shutdown_d	1987
5.412.2.36	cl_statchkdir_d	1987
5.412.2.37	cl_statfree_d	1987

5.412.2.38	cl_statindir_d	1987
5.412.2.39	cl_strerror_d	1987
5.412.2.40	COMP_zlib_cleanup_d	1987
5.412.2.41	CONF_modules_unload_d	1988
5.412.2.42	CRYPTO_cleanup_all_ex_data_d	1988
5.412.2.43	CRYPTO_free_d	1988
5.412.2.44	CRYPTO_num_locks_d	1988
5.412.2.45	DH_check_d	1988
5.412.2.46	DH_free_d	1988
5.412.2.47	DH_new_d	1988
5.412.2.48	dkim_body_d	1988
5.412.2.49	dkim_chunk_d	1988
5.412.2.50	dkim_close_d	1988
5.412.2.51	dkim_eoh_d	1988
5.412.2.52	dkim_eom_d	1988
5.412.2.53	dkim_free_d	1989
5.412.2.54	dkim_geterror_d	1989
5.412.2.55	dkim_getresultstr_d	1989
5.412.2.56	dkim_getsighdrx_d	1989
5.412.2.57	dkim_header_d	1989
5.412.2.58	dkim_init_d	1989
5.412.2.59	dkim_libversion_d	1989
5.412.2.60	dkim_mfree_d	1989
5.412.2.61	dkim_sign_d	1989
5.412.2.62	dkim_test_dns_put_d	1989
5.412.2.63	dkim_verify_d	1989
5.412.2.64	dspam_attach_d	1989
5.412.2.65	dspam_create_d	1989
5.412.2.66	dspam_destroy_d	1990
5.412.2.67	dspam_detach_d	1990
5.412.2.68	dspam_init_driver_d	1990
5.412.2.69	dspam_process_d	1990
5.412.2.70	dspam_shutdown_driver_d	1990
5.412.2.71	dspam_version_d	1990
5.412.2.72	EC_GROUP_clear_free_d	1990
5.412.2.73	EC_GROUP_free_d	1990
5.412.2.74	EC_GROUP_new_by_curve_name_d	1990
5.412.2.75	EC_GROUP_precompute_mult_d	1990
5.412.2.76	EC_KEY_check_key_d	1990

5.412.2.77	EC_KEY_free_d	1991
5.412.2.78	EC_KEY_generate_key_d	1991
5.412.2.79	EC_KEY_get0_group_d	1991
5.412.2.80	EC_KEY_get0_private_key_d	1991
5.412.2.81	EC_KEY_get0_public_key_d	1991
5.412.2.82	EC_KEY_new_by_curve_name_d	1991
5.412.2.83	EC_KEY_new_d	1991
5.412.2.84	EC_KEY_set_group_d	1991
5.412.2.85	EC_KEY_set_private_key_d	1991
5.412.2.86	EC_POINT_free_d	1991
5.412.2.87	EC_POINT_new_d	1992
5.412.2.88	ECDSA_SIG_free_d	1992
5.412.2.89	ENGINE_cleanup_d	1992
5.412.2.90	ERR_clear_error_d	1992
5.412.2.91	ERR_free_strings_d	1992
5.412.2.92	ERR_get_error_d	1992
5.412.2.93	ERR_load_crypto_strings_d	1992
5.412.2.94	ERR_peek_error_d	1992
5.412.2.95	ERR_print_errors_fp_d	1992
5.412.2.96	ERR_remove_thread_state_d	1992
5.412.2.97	EVP_aes_256_cbc_d	1992
5.412.2.98	EVP_aes_256_gcm_d	1992
5.412.2.99	EVP_CIPHER_block_size_d	1993
5.412.2.100	EVP_CIPHER_CTX_block_size_d	1993
5.412.2.101	EVP_CIPHER_CTX_cleanup_d	1993
5.412.2.102	EVP_CIPHER_CTX_free_d	1993
5.412.2.103	EVP_CIPHER_CTX_init_d	1993
5.412.2.104	EVP_CIPHER_CTX_iv_length_d	1993
5.412.2.105	EVP_CIPHER_CTX_key_length_d	1993
5.412.2.106	EVP_CIPHER_CTX_new_d	1993
5.412.2.107	EVP_CIPHER_CTX_set_padding_d	1993
5.412.2.108	EVP_CIPHER_flags_d	1993
5.412.2.109	EVP_CIPHER_iv_length_d	1993
5.412.2.110	EVP_CIPHER_key_length_d	1994
5.412.2.111	EVP_CIPHER_nid_d	1994
5.412.2.112	EVP_cleanup_d	1994
5.412.2.113	EVP_DigestInit_d	1994
5.412.2.114	EVP_get_cipherbyname_d	1994
5.412.2.115	EVP_get_digestbyname_d	1994

5.412.2.11	EV VP_md4_d	1994
5.412.2.11	EV VP_md5_d	1994
5.412.2.11	EV VP_MD_CTX_cleanup_d	1994
5.412.2.11	EV VP_MD_CTX_init_d	1994
5.412.2.12	EV VP_MD_size_d	1994
5.412.2.12	EV VP_MD_type_d	1994
5.412.2.12	EV VP_PKEY_new_d	1995
5.412.2.12	EV VP_PKEY_set1_RSA_d	1995
5.412.2.12	EV VP_ripemd160_d	1995
5.412.2.12	EV VP_sha1_d	1995
5.412.2.12	EV VP_sha224_d	1995
5.412.2.12	EV VP_sha256_d	1995
5.412.2.12	EV VP_sha384_d	1995
5.412.2.12	EV VP_sha512_d	1995
5.412.2.13	EV VP_sha_d	1995
5.412.2.13	FT_Done_FreeType_d	1995
5.412.2.13	FT_Init_FreeType_d	1995
5.412.2.13	FT_Library_Version_d	1995
5.412.2.13	4dFree_d	1996
5.412.2.13	5dImageColorResolve_d	1996
5.412.2.13	6dImageCreate_d	1996
5.412.2.13	7dImageDestroy_d	1996
5.412.2.13	8dImageGifPtr_d	1996
5.412.2.13	9dImageJpegPtr_d	1996
5.412.2.14	0dImageSetPixel_d	1996
5.412.2.14	1dImageStringFT_d	1996
5.412.2.14	2dVersionString_d	1996
5.412.2.14	EV MAC_CTX_cleanup_d	1996
5.412.2.14	EV MAC_CTX_init_d	1996
5.412.2.14	3d_ECPrivateKey_d	1997
5.412.2.14	4d_OCSP_CERTID_d	1997
5.412.2.14	7d_X509_d	1997
5.412.2.14	8o_ECPublicKey_d	1997
5.412.2.14	9peg_version_d	1997
5.412.2.15	0_dlexit_d	1997
5.412.2.15	1x_1_compress_d	1997
5.412.2.15	2x_decompress_safe_d	1997
5.412.2.15	3o_adler32_d	1997
5.412.2.15	4o_version_string_d	1997

5.412.2.155	memcached_add_d	1997
5.412.2.156	memcached_append_d	1997
5.412.2.157	memcached_behavior_set_d	1997
5.412.2.158	memcached_cas_d	1998
5.412.2.159	memcached_create_d	1998
5.412.2.160	memcached_decrement_d	1998
5.412.2.161	memcached_decrement_with_initial_d	1998
5.412.2.162	memcached_delete_d	1998
5.412.2.163	memcached_flush_d	1998
5.412.2.164	memcached_free_d	1998
5.412.2.165	memcached_get_d	1998
5.412.2.166	memcached_increment_d	1998
5.412.2.167	memcached_increment_with_initial_d	1998
5.412.2.168	memcached_lib_version_d	1998
5.412.2.169	memcached_prepend_d	1999
5.412.2.170	memcached_replace_d	1999
5.412.2.171	memcached_server_add_with_weight_d	1999
5.412.2.172	memcached_set_d	1999
5.412.2.173	memcached_strerror_d	1999
5.412.2.174	my_once_free_d	1999
5.412.2.175	mysql_affected_rows_d	1999
5.412.2.176	mysql_character_set_name_d	1999
5.412.2.177	mysql_close_d	1999
5.412.2.178	mysql_embedded_d	1999
5.412.2.179	mysql_errno_d	1999
5.412.2.180	mysql_error_d	1999
5.412.2.181	mysql_escape_string_d	2000
5.412.2.182	mysql_fetch_field_d	2000
5.412.2.183	mysql_fetch_row_d	2000
5.412.2.184	mysql_free_result_d	2000
5.412.2.185	mysql_get_client_version_d	2000
5.412.2.186	mysql_get_server_info_d	2000
5.412.2.187	mysql_init_d	2000
5.412.2.188	mysql_insert_id_d	2000
5.412.2.189	mysql_num_fields_d	2000
5.412.2.190	mysql_num_rows_d	2000
5.412.2.191	mysql_options_d	2000
5.412.2.192	mysql_ping_d	2000
5.412.2.193	mysql_real_connect_d	2000

5.412.2.194	mysql_real_query_d	2001
5.412.2.195	mysql_server_end_d	2001
5.412.2.196	mysql_server_init_d	2001
5.412.2.197	mysql_set_character_set_d	2001
5.412.2.198	mysql_stmt_affected_rows_d	2001
5.412.2.199	mysql_stmt_attr_set_d	2001
5.412.2.200	mysql_stmt_bind_param_d	2001
5.412.2.201	mysql_stmt_bind_result_d	2001
5.412.2.202	mysql_stmt_close_d	2001
5.412.2.203	mysql_stmt_errno_d	2001
5.412.2.204	mysql_stmt_error_d	2001
5.412.2.205	mysql_stmt_execute_d	2001
5.412.2.206	mysql_stmt_fetch_d	2002
5.412.2.207	mysql_stmt_free_result_d	2002
5.412.2.208	mysql_stmt_init_d	2002
5.412.2.209	mysql_stmt_insert_id_d	2002
5.412.2.210	mysql_stmt_num_rows_d	2002
5.412.2.211	mysql_stmt_prepare_d	2002
5.412.2.212	mysql_stmt_reset_d	2002
5.412.2.213	mysql_stmt_result_metadata_d	2002
5.412.2.214	mysql_stmt_store_result_d	2002
5.412.2.215	mysql_store_result_d	2002
5.412.2.216	mysql_thread_end_d	2002
5.412.2.217	mysql_thread_id_d	2002
5.412.2.218	mysql_thread_init_d	2003
5.412.2.219	mysql_thread_safe_d	2003
5.412.2.220	OBJ_cleanup_d	2003
5.412.2.221	OBJ_NAME_cleanup_d	2003
5.412.2.222	OBJ_nid2sn_d	2003
5.412.2.223	CSP_BASICRESP_free_d	2003
5.412.2.224	CSP_check_nonce_d	2003
5.412.2.225	CSP_REQ_CTX_set1_req_d	2003
5.412.2.226	CSP_request_add0_id_d	2003
5.412.2.227	CSP_REQUEST_free_d	2003
5.412.2.228	CSP_REQUEST_new_d	2003
5.412.2.229	CSP_RESPONSE_free_d	2003
5.412.2.230	CSP_response_get1_basic_d	2004
5.412.2.231	CSP_response_status_d	2004
5.412.2.232	CSP_response_status_str_d	2004

5.412.2.23	OPENSSL_add_all_algorithms_noconf_d	2004
5.412.2.23	RAND_bytes_d	2004
5.412.2.23	RAND_cleanup_d	2004
5.412.2.23	RAND_load_file_d	2004
5.412.2.23	RAND_status_d	2004
5.412.2.23	RSA_free_d	2004
5.412.2.23	RSA_new_d	2004
5.412.2.24	RSA_public_key_dup_d	2004
5.412.2.24	SHA1_Final_d	2004
5.412.2.24	SHA1_Init_d	2005
5.412.2.24	SHA1_Update_d	2005
5.412.2.24	SHA256_Final_d	2005
5.412.2.24	SHA256_Init_d	2005
5.412.2.24	SHA512_Final_d	2005
5.412.2.24	SHA512_Init_d	2005
5.412.2.24	sk_num_d	2005
5.412.2.24	sk_pop_d	2005
5.412.2.25	sk_pop_free_d	2005
5.412.2.25	sk_value_d	2005
5.412.2.25	SSL_accept_d	2005
5.412.2.25	SSL_CIPHER_get_name_d	2005
5.412.2.25	SSL_CIPHER_get_version_d	2006
5.412.2.25	SSL_connect_d	2006
5.412.2.25	SSL_ctrl_d	2006
5.412.2.25	SSL_CTX_callback_ctrl_d	2006
5.412.2.25	SSL_CTX_check_private_key_d	2006
5.412.2.25	SSL_CTX_free_d	2006
5.412.2.26	SSL_CTX_new_d	2006
5.412.2.26	SSL_CTX_set_cipher_list_d	2006
5.412.2.26	SSL_do_handshake_d	2006
5.412.2.26	SSL_free_d	2006
5.412.2.26	SSL_get_current_cipher_d	2006
5.412.2.26	SSL_get_error_d	2006
5.412.2.26	SSL_get_fd_d	2006
5.412.2.26	SSL_get_peer_certificate_d	2007
5.412.2.26	SSL_get_read_ahead_d	2007
5.412.2.26	SSL_get_rfd_d	2007
5.412.2.27	SSL_get_shutdown_d	2007
5.412.2.27	SSL_get_version_d	2007

5.412.2.27	SSL_get_wbio_d	2007
5.412.2.27	SSL_library_init_d	2007
5.412.2.27	SSL_load_error_strings_d	2007
5.412.2.27	SSL_new_d	2007
5.412.2.27	SSL_peek_d	2007
5.412.2.27	SSL_pending_d	2007
5.412.2.27	SSL_read_d	2007
5.412.2.27	SSL_set_accept_state_d	2007
5.412.2.28	SSL_set_bio_d	2007
5.412.2.28	SSL_set_connect_state_d	2008
5.412.2.28	SSL_set_fd_d	2008
5.412.2.28	SSL_set_read_ahead_d	2008
5.412.2.28	SSL_shutdown_d	2008
5.412.2.28	SSL_version_str_d	2008
5.412.2.28	SSL_want_d	2008
5.412.2.28	SSL_write_d	2008
5.412.2.28	SSLey_version_d	2008
5.412.2.28	SSLv23_client_method_d	2008
5.412.2.29	SSLv23_server_method_d	2008
5.412.2.29	TLSv1_server_method_d	2008
5.412.2.29	X509_check_issued_d	2008
5.412.2.29	X509_email_free_d	2008
5.412.2.29	X509_get1_ocsp_d	2008
5.412.2.29	X509_get_ext_count_d	2009
5.412.2.29	X509_get_ext_d	2009
5.412.2.29	X509_get_subject_name_d	2009
5.412.2.29	X509_LOOKUP_file_d	2009
5.412.2.29	X509_NAME_online_d	2009
5.412.2.30	X509_STORE_CTX_free_d	2009
5.412.2.30	X509_STORE_CTX_get_error_d	2009
5.412.2.30	X509_STORE_CTX_get_error_depth_d	2009
5.412.2.30	X509_STORE_CTX_new_d	2009
5.412.2.30	X509_STORE_free_d	2009
5.412.2.30	X509_STORE_new_d	2009
5.412.2.30	X509_verify_cert_d	2009
5.412.2.30	X509_verify_cert_error_string_d	2009
5.413	src/providers/symbols.h File Reference	2010
5.413.1	Define Documentation	2023
5.413.1.1	CONFIG_DEFAULT	2023

5.413.1.2	lint	2023
5.413.1.3	LOGDIR	2023
5.413.1.4	M_BIND	2023
5.413.2	Function Documentation	2023
5.413.2.1	STACK_OF	2023
5.413.3	Variable Documentation	2023
5.413.3.1	__lzo_init_v2_d	2023
5.413.3.2	ASN1_GENERALIZEDTIME_print_d	2023
5.413.3.3	ASN1_INTEGER_to_BN_d	2023
5.413.3.4	ASN1_STRING_data_d	2024
5.413.3.5	ASN1_STRING_TABLE_cleanup_d	2024
5.413.3.6	BIO_free_all_d	2024
5.413.3.7	BIO_free_d	2024
5.413.3.8	BIO_new_fp_d	2024
5.413.3.9	BIO_new_socket_d	2024
5.413.3.10	BIO_sock_cleanup_d	2024
5.413.3.11	BIO_vprintf_d	2024
5.413.3.12	BN_bin2bn_d	2024
5.413.3.13	BN_bn2bin_d	2024
5.413.3.14	BN_bn2dec_d	2024
5.413.3.15	BN_bn2hex_d	2024
5.413.3.16	BN_bn2mpi_d	2025
5.413.3.17	BN_cmp_d	2025
5.413.3.18	BN_CTX_free_d	2025
5.413.3.19	BN_CTX_new_d	2025
5.413.3.20	BN_CTX_start_d	2025
5.413.3.21	BN_free_d	2025
5.413.3.22	BN_hex2bn_d	2025
5.413.3.23	BN_mpi2bn_d	2025
5.413.3.24	BN_num_bits_d	2025
5.413.3.25	BUF_strlcat_d	2025
5.413.3.26	BZ2_bzBuffToBuffCompress_d	2025
5.413.3.27	BZ2_bzBuffToBuffDecompress_d	2025
5.413.3.28	BZ2_bzlibVersion_d	2025
5.413.3.29	cl_countsigs_d	2026
5.413.3.30	cl_engine_compile_d	2026
5.413.3.31	cl_engine_free_d	2026
5.413.3.32	cl_engine_new_d	2026
5.413.3.33	cl_engine_set_num_d	2026

5.413.3.34cl_engine_set_str_d	2026
5.413.3.35cl_init_d	2026
5.413.3.36cl_load_d	2026
5.413.3.37cl_retver_d	2026
5.413.3.38cl_scandesc_d	2026
5.413.3.39cl_shutdown_d	2026
5.413.3.40cl_statchkdir_d	2026
5.413.3.41cl_statfree_d	2027
5.413.3.42cl_statinidir_d	2027
5.413.3.43cl_strerror_d	2027
5.413.3.44COMP_zlib_cleanup_d	2027
5.413.3.45compress2_d	2027
5.413.3.46compressBound_d	2027
5.413.3.47CONF_modules_unload_d	2027
5.413.3.48CRYPTO_cleanup_all_ex_data_d	2027
5.413.3.49CRYPTO_free_d	2027
5.413.3.50CRYPTO_num_locks_d	2027
5.413.3.51CRYPTO_set_id_callback_d	2027
5.413.3.52CRYPTO_set_locked_mem_functions_d	2027
5.413.3.53CRYPTO_set_locking_callback_d	2028
5.413.3.54CRYPTO_set_mem_functions_d	2028
5.413.3.55d2i_ECDSA_SIG_d	2028
5.413.3.56d2i_ECPrivateKey_d	2028
5.413.3.57d2i_OCSP_RESPONSE_d	2028
5.413.3.58DH_check_d	2028
5.413.3.59DH_free_d	2028
5.413.3.60DH_generate_parameters_ex_d	2028
5.413.3.61DH_new_d	2028
5.413.3.62dkim_body_d	2028
5.413.3.63dkim_chunk_d	2028
5.413.3.64dkim_close_d	2028
5.413.3.65dkim_eoh_d	2028
5.413.3.66dkim_eom_d	2029
5.413.3.67dkim_free_d	2029
5.413.3.68dkim_geterror_d	2029
5.413.3.69dkim_getresultstr_d	2029
5.413.3.70dkim_getsighdrx_d	2029
5.413.3.71dkim_header_d	2029
5.413.3.72dkim_init_d	2029

5.413.3.73	dkim_libversion_d	2029
5.413.3.74	dkim_mfree_d	2029
5.413.3.75	dkim_sign_d	2029
5.413.3.76	dkim_test_dns_put_d	2029
5.413.3.77	dkim_verify_d	2029
5.413.3.78	dspam_attach_d	2030
5.413.3.79	dspam_create_d	2030
5.413.3.80	dspam_destroy_d	2030
5.413.3.81	dspam_detach_d	2030
5.413.3.82	dspam_init_driver_d	2030
5.413.3.83	dspam_process_d	2030
5.413.3.84	dspam_shutdown_driver_d	2030
5.413.3.85	dspam_version_d	2030
5.413.3.86	EC_GROUP_clear_free_d	2030
5.413.3.87	EC_GROUP_free_d	2030
5.413.3.88	EC_GROUP_new_by_curve_name_d	2030
5.413.3.89	EC_GROUP_precompute_mult_d	2031
5.413.3.90	EC_GROUP_set_point_conversion_form_d	2031
5.413.3.91	EC_KEY_check_key_d	2031
5.413.3.92	EC_KEY_free_d	2031
5.413.3.93	EC_KEY_generate_key_d	2031
5.413.3.94	EC_KEY_get0_group_d	2031
5.413.3.95	EC_KEY_get0_private_key_d	2031
5.413.3.96	EC_KEY_get0_public_key_d	2031
5.413.3.97	EC_KEY_get_conv_form_d	2031
5.413.3.98	EC_KEY_new_by_curve_name_d	2031
5.413.3.99	EC_KEY_new_d	2032
5.413.3.100	EC_KEY_set_conv_form_d	2032
5.413.3.101	EC_KEY_set_group_d	2032
5.413.3.102	EC_KEY_set_private_key_d	2032
5.413.3.103	EC_KEY_set_public_key_d	2032
5.413.3.104	EC_POINT_cmp_d	2032
5.413.3.105	EC_POINT_free_d	2032
5.413.3.106	EC_POINT_hex2point_d	2032
5.413.3.107	EC_POINT_mul_d	2032
5.413.3.108	EC_POINT_new_d	2032
5.413.3.109	EC_POINT_oct2point_d	2032
5.413.3.110	EC_POINT_point2hex_d	2032
5.413.3.111	EC_POINT_point2oct_d	2033

5.413.3.11ECDH_compute_key_d	2033
5.413.3.11BCDSA_do_sign_d	2033
5.413.3.11BCDSA_do_verify_d	2033
5.413.3.11BCDSA_SIG_free_d	2033
5.413.3.11BD25519_keypair_d	2033
5.413.3.11BD25519_keypair_from_seed_d	2033
5.413.3.11BD25519_sign_d	2033
5.413.3.11BD25519_verify_d	2033
5.413.3.12ENGINE_cleanup_d	2033
5.413.3.12ERR_clear_error_d	2033
5.413.3.12ERR_error_string_n_d	2034
5.413.3.12ERR_free_strings_d	2034
5.413.3.12ERR_get_error_d	2034
5.413.3.12ERR_load_crypto_strings_d	2034
5.413.3.12ERR_peek_error_d	2034
5.413.3.12ERR_peek_error_line_data_d	2034
5.413.3.12ERR_print_errors_fp_d	2034
5.413.3.12ERR_put_error_d	2034
5.413.3.13ERR_remove_thread_state_d	2034
5.413.3.13EVP_aes_256_cbc_d	2034
5.413.3.13EVP_aes_256_gcm_d	2034
5.413.3.13EVP_CIPHER_block_size_d	2034
5.413.3.13EVP_CIPHER_CTX_block_size_d	2034
5.413.3.13EVP_CIPHER_CTX_cleanup_d	2035
5.413.3.13EVP_CIPHER_CTX_ctrl_d	2035
5.413.3.13EVP_CIPHER_CTX_flags_d	2035
5.413.3.13EVP_CIPHER_CTX_free_d	2035
5.413.3.13EVP_CIPHER_CTX_init_d	2035
5.413.3.14EVP_CIPHER_CTX_iv_length_d	2035
5.413.3.14EVP_CIPHER_CTX_key_length_d	2035
5.413.3.14EVP_CIPHER_CTX_new_d	2035
5.413.3.14EVP_CIPHER_CTX_set_padding_d	2035
5.413.3.14EVP_CIPHER_flags_d	2035
5.413.3.14EVP_CIPHER_iv_length_d	2035
5.413.3.14EVP_CIPHER_key_length_d	2036
5.413.3.14EVP_CIPHER_nid_d	2036
5.413.3.14EVP_cleanup_d	2036
5.413.3.14EVP_DecryptFinal_ex_d	2036
5.413.3.15EVP_DecryptInit_ex_d	2036

5.413.3.151EVP_DecryptUpdate_d	2036
5.413.3.152EVP_Digest_d	2036
5.413.3.153EVP_DigestFinal_d	2036
5.413.3.154EVP_DigestFinal_ex_d	2036
5.413.3.155EVP_DigestInit_d	2036
5.413.3.156EVP_DigestInit_ex_d	2036
5.413.3.157EVP_DigestUpdate_d	2036
5.413.3.158EVP_EncryptFinal_ex_d	2037
5.413.3.159EVP_EncryptInit_ex_d	2037
5.413.3.160EVP_EncryptUpdate_d	2037
5.413.3.161EVP_get_cipherbyname_d	2037
5.413.3.162EVP_get_digestbyname_d	2037
5.413.3.163EVP_md4_d	2037
5.413.3.164EVP_md5_d	2037
5.413.3.165EVP_MD_CTX_cleanup_d	2037
5.413.3.166EVP_MD_CTX_init_d	2037
5.413.3.167EVP_MD_size_d	2037
5.413.3.168EVP_MD_type_d	2037
5.413.3.169EVP_PKEY_new_d	2038
5.413.3.170EVP_PKEY_set1_RSA_d	2038
5.413.3.171EVP_ripemd160_d	2038
5.413.3.172EVP_sha1_d	2038
5.413.3.173EVP_sha224_d	2038
5.413.3.174EVP_sha256_d	2038
5.413.3.175EVP_sha384_d	2038
5.413.3.176EVP_sha512_d	2038
5.413.3.177EVP_sha_d	2038
5.413.3.178EVP_VerifyFinal_d	2038
5.413.3.179EVT_Done_FreeType_d	2038
5.413.3.180EVT_Init_FreeType_d	2038
5.413.3.181EVT_Library_Version_d	2039
5.413.3.182EVT_Free_d	2039
5.413.3.183EVT_ImageColorResolve_d	2039
5.413.3.184EVT_ImageCreate_d	2039
5.413.3.185EVT_ImageDestroy_d	2039
5.413.3.186EVT_ImageGifPtr_d	2039
5.413.3.187EVT_ImageJpegPtr_d	2039
5.413.3.188EVT_ImageSetPixel_d	2039
5.413.3.189EVT_ImageStringFT_d	2039

5.413.3.190dVersionString_d	2039
5.413.3.191HMAC_CTX_cleanup_d	2039
5.413.3.192HMAC_CTX_init_d	2039
5.413.3.193HMAC_Final_d	2040
5.413.3.194HMAC_Init_ex_d	2040
5.413.3.195HMAC_Update_d	2040
5.413.3.196d_ECDSA_SIG_d	2040
5.413.3.197d_ECPrivateKey_d	2040
5.413.3.198d_OCSP_CERTID_d	2040
5.413.3.199d_OCSP_RESPONSE_d	2040
5.413.3.200d_X509_d	2040
5.413.3.201o_ECPublicKey_d	2040
5.413.3.202ansson_version_d	2040
5.413.3.203peg_version_d	2040
5.413.3.204on_array_append_d	2041
5.413.3.205on_array_append_new_d	2041
5.413.3.206on_array_clear_d	2041
5.413.3.207on_array_d	2041
5.413.3.208on_array_extend_d	2041
5.413.3.209on_array_get_d	2041
5.413.3.210on_array_insert_d	2041
5.413.3.211on_array_insert_new_d	2041
5.413.3.212on_array_remove_d	2041
5.413.3.213on_array_set_d	2041
5.413.3.214on_array_set_new_d	2041
5.413.3.215on_array_size_d	2041
5.413.3.216on_copy_d	2041
5.413.3.217on_decref_d	2041
5.413.3.218on_deep_copy_d	2042
5.413.3.219on_delete_d	2042
5.413.3.220on_dump_file_d	2042
5.413.3.221on_dumpf_d	2042
5.413.3.222on_dumps_d	2042
5.413.3.223on_equal_d	2042
5.413.3.224on_false_d	2042
5.413.3.225on_incref_d	2042
5.413.3.226on_integer_d	2042
5.413.3.227on_integer_set_d	2042
5.413.3.228on_integer_value_d	2042

5.413.3.229	son_load_file_d	2042
5.413.3.230	son_loadf_d	2042
5.413.3.231	son_loads_d	2042
5.413.3.232	son_null_d	2042
5.413.3.233	son_number_value_d	2042
5.413.3.234	son_object_clear_d	2042
5.413.3.235	son_object_d	2042
5.413.3.236	son_object_del_d	2043
5.413.3.237	son_object_get_d	2043
5.413.3.238	son_object_iter_at_d	2043
5.413.3.239	son_object_iter_d	2043
5.413.3.240	son_object_iter_key_d	2043
5.413.3.241	son_object_iter_next_d	2043
5.413.3.242	son_object_iter_set_d	2043
5.413.3.243	son_object_iter_set_new_d	2043
5.413.3.244	son_object_iter_value_d	2043
5.413.3.245	son_object_set_d	2043
5.413.3.246	son_object_set_new_d	2043
5.413.3.247	son_object_set_new_nocheck_d	2043
5.413.3.248	son_object_set_nocheck_d	2043
5.413.3.249	son_object_size_d	2043
5.413.3.250	son_object_update_d	2044
5.413.3.251	son_pack_d	2044
5.413.3.252	son_pack_ex_d	2044
5.413.3.253	son_real_d	2044
5.413.3.254	son_real_set_d	2044
5.413.3.255	son_real_value_d	2044
5.413.3.256	son_set_alloc_funcs_d	2044
5.413.3.257	son_string_d	2044
5.413.3.258	son_string_nocheck_d	2044
5.413.3.259	son_string_set_d	2044
5.413.3.260	son_string_set_nocheck_d	2044
5.413.3.261	son_string_value_d	2044
5.413.3.262	son_true_d	2044
5.413.3.263	son_type_string_d	2044
5.413.3.264	son_unpack_d	2044
5.413.3.265	son_unpack_ex_d	2044
5.413.3.266	son_vpack_ex_d	2045
5.413.3.267	son_vunpack_ex_d	2045

5.413.3.268	dlexit_d	2045
5.413.3.269	zlib_deflate_d	2045
5.413.3.270	zlib_deflate_quick_d	2045
5.413.3.271	zlib_deflate_set_d	2045
5.413.3.272	zlib_inflate_d	2045
5.413.3.273	zlib_inflate_quick_d	2045
5.413.3.274	zlib_inflate_set_d	2045
5.413.3.275	zlib_memcached_behavior_set_d	2045
5.413.3.276	zlib_memcached_cas_d	2045
5.413.3.277	zlib_memcached_create_d	2045
5.413.3.278	zlib_memcached_decrement_d	2046
5.413.3.279	zlib_memcached_decrement_with_initial_d	2046
5.413.3.280	zlib_memcached_delete_d	2046
5.413.3.281	zlib_memcached_flush_d	2046
5.413.3.282	zlib_memcached_free_d	2046
5.413.3.283	zlib_memcached_get_d	2046
5.413.3.284	zlib_memcached_increment_d	2046
5.413.3.285	zlib_memcached_increment_with_initial_d	2046
5.413.3.286	zlib_memcached_lib_version_d	2046
5.413.3.287	zlib_memcached_prepend_d	2046
5.413.3.288	zlib_memcached_replace_d	2046
5.413.3.289	zlib_memcached_server_add_with_weight_d	2046
5.413.3.290	zlib_memcached_set_d	2047
5.413.3.291	zlib_memcached_strerror_d	2047
5.413.3.292	zlib_once_free_d	2047
5.413.3.293	mysql_affected_rows_d	2047
5.413.3.294	mysql_character_set_name_d	2047
5.413.3.295	mysql_close_d	2047
5.413.3.296	mysql_embedded_d	2047
5.413.3.297	mysql_errno_d	2047
5.413.3.298	mysql_error_d	2047
5.413.3.299	mysql_escape_string_d	2047
5.413.3.300	mysql_fetch_field_d	2047
5.413.3.301	mysql_fetch_row_d	2047
5.413.3.302	mysql_free_result_d	2047
5.413.3.303	mysql_get_client_version_d	2048
5.413.3.304	mysql_get_server_info_d	2048
5.413.3.305	mysql_init_d	2048
5.413.3.306	mysql_insert_id_d	2048

5.413.3.307	mysql_num_fields_d	2048
5.413.3.308	mysql_num_rows_d	2048
5.413.3.309	mysql_options_d	2048
5.413.3.310	mysql_ping_d	2048
5.413.3.311	mysql_real_connect_d	2048
5.413.3.312	mysql_real_query_d	2048
5.413.3.313	mysql_server_end_d	2048
5.413.3.314	mysql_server_init_d	2048
5.413.3.315	mysql_set_character_set_d	2049
5.413.3.316	mysql_stmt_affected_rows_d	2049
5.413.3.317	mysql_stmt_attr_set_d	2049
5.413.3.318	mysql_stmt_bind_param_d	2049
5.413.3.319	mysql_stmt_bind_result_d	2049
5.413.3.320	mysql_stmt_close_d	2049
5.413.3.321	mysql_stmt_errno_d	2049
5.413.3.322	mysql_stmt_error_d	2049
5.413.3.323	mysql_stmt_execute_d	2049
5.413.3.324	mysql_stmt_fetch_d	2049
5.413.3.325	mysql_stmt_free_result_d	2049
5.413.3.326	mysql_stmt_init_d	2049
5.413.3.327	mysql_stmt_insert_id_d	2049
5.413.3.328	mysql_stmt_num_rows_d	2050
5.413.3.329	mysql_stmt_prepare_d	2050
5.413.3.330	mysql_stmt_reset_d	2050
5.413.3.331	mysql_stmt_result_metadata_d	2050
5.413.3.332	mysql_stmt_store_result_d	2050
5.413.3.333	mysql_store_result_d	2050
5.413.3.334	mysql_thread_end_d	2050
5.413.3.335	mysql_thread_id_d	2050
5.413.3.336	mysql_thread_init_d	2050
5.413.3.337	mysql_thread_safe_d	2050
5.413.3.338	i_ECPublicKey_d	2050
5.413.3.339	OBJ_cleanup_d	2050
5.413.3.340	OBJ_NAME_cleanup_d	2051
5.413.3.341	OBJ_nid2sn_d	2051
5.413.3.342	CSP_basic_verify_d	2051
5.413.3.343	CSP_BASICRESP_free_d	2051
5.413.3.344	CSP_cert_to_id_d	2051
5.413.3.345	CSP_check_nonce_d	2051

5.413.3.340	CSP_check_validity_d	2051
5.413.3.340	CSP_parse_url_d	2051
5.413.3.340	CSP_REQ_CTX_add1_header_d	2051
5.413.3.340	CSP_REQ_CTX_set1_req_d	2051
5.413.3.350	CSP_request_add0_id_d	2051
5.413.3.350	CSP_request_add1_nonce_d	2051
5.413.3.350	CSP_REQUEST_free_d	2052
5.413.3.350	CSP_REQUEST_new_d	2052
5.413.3.350	CSP_REQUEST_print_d	2052
5.413.3.350	CSP_resp_find_status_d	2052
5.413.3.350	CSP_RESPONSE_free_d	2052
5.413.3.350	CSP_response_get1_basic_d	2052
5.413.3.350	CSP_RESPONSE_print_d	2052
5.413.3.350	CSP_response_status_d	2052
5.413.3.360	CSP_response_status_str_d	2052
5.413.3.360	CSP_sendreq_nbio_d	2052
5.413.3.360	CSP_sendreq_new_d	2052
5.413.3.360	OPENSSL_add_all_algorithms_noconf_d	2052
5.413.3.360	png_access_version_number_d	2053
5.413.3.360	RAND_bytes_d	2053
5.413.3.360	RAND_cleanup_d	2053
5.413.3.360	RAND_load_file_d	2053
5.413.3.360	RAND_status_d	2053
5.413.3.360	RSA_free_d	2053
5.413.3.370	RSA_new_d	2053
5.413.3.370	RSAPublicKey_dup_d	2053
5.413.3.370	SHA1_Final_d	2053
5.413.3.370	SHA1_Init_d	2053
5.413.3.370	SHA1_Update_d	2053
5.413.3.370	SHA256_Final_d	2054
5.413.3.370	SHA256_Init_d	2054
5.413.3.370	SHA256_Update_d	2054
5.413.3.370	SHA512_d	2054
5.413.3.370	SHA512_Final_d	2054
5.413.3.380	SHA512_Init_d	2054
5.413.3.380	SHA512_Update_d	2054
5.413.3.380	sk_num_d	2054
5.413.3.380	sk_pop_d	2054
5.413.3.380	sk_pop_free_d	2054

5.413.3.385	sk_value_d	2054
5.413.3.386	SPF_dns_zone_add_str_d	2054
5.413.3.387	SPF_dns_zone_new_d	2054
5.413.3.388	SPF_get_lib_version_d	2054
5.413.3.389	SPF_request_free_d	2055
5.413.3.390	SPF_request_new_d	2055
5.413.3.391	SPF_request_query_mailfrom_d	2055
5.413.3.392	SPF_request_set_env_from_d	2055
5.413.3.393	SPF_request_set_helo_dom_d	2055
5.413.3.394	SPF_request_set_ipv4_d	2055
5.413.3.395	SPF_request_set_ipv6_d	2055
5.413.3.396	SPF_response_free_d	2055
5.413.3.397	SPF_response_reason_d	2055
5.413.3.398	SPF_response_result_d	2055
5.413.3.399	SPF_server_free_d	2055
5.413.3.400	SPF_server_new_d	2055
5.413.3.401	SPF_strerror_d	2056
5.413.3.402	SPF_strerror_reason_d	2056
5.413.3.403	SPF_strerror_result_d	2056
5.413.3.404	SSL_accept_d	2056
5.413.3.405	SSL_CIPHER_get_bits_d	2056
5.413.3.406	SSL_CIPHER_get_name_d	2056
5.413.3.407	SSL_CIPHER_get_version_d	2056
5.413.3.408	SSL_connect_d	2056
5.413.3.409	SSL_ctrl_d	2056
5.413.3.410	SSL_CTX_callback_ctrl_d	2056
5.413.3.411	SSL_CTX_check_private_key_d	2056
5.413.3.412	SSL_CTX_ctrl_d	2056
5.413.3.413	SSL_CTX_free_d	2056
5.413.3.414	SSL_CTX_load_verify_locations_d	2057
5.413.3.415	SSL_CTX_new_d	2057
5.413.3.416	SSL_CTX_set_cipher_list_d	2057
5.413.3.417	SSL_CTX_set_tmp_dh_callback_d	2057
5.413.3.418	SSL_CTX_set_tmp_ecdh_callback_d	2057
5.413.3.419	SSL_CTX_set_verify_d	2057
5.413.3.420	SSL_CTX_use_certificate_chain_file_d	2057
5.413.3.421	SSL_CTX_use_PrivateKey_file_d	2057
5.413.3.422	SSL_do_handshake_d	2057
5.413.3.423	SSL_free_d	2057

5.413.3.42	SSL_get_current_cipher_d	2057
5.413.3.42	SSL_get_error_d	2057
5.413.3.42	SSL_get_fd_d	2057
5.413.3.42	SSL_get_peer_cert_chain_d	2058
5.413.3.42	SSL_get_peer_certificate_d	2058
5.413.3.42	SSL_get_read_ahead_d	2058
5.413.3.43	SSL_get_rfd_d	2058
5.413.3.43	SSL_get_shutdown_d	2058
5.413.3.43	SSL_get_version_d	2058
5.413.3.43	SSL_get_wbio_d	2058
5.413.3.43	SSL_library_init_d	2058
5.413.3.43	SSL_load_error_strings_d	2058
5.413.3.43	SSL_new_d	2058
5.413.3.43	SSL_peek_d	2058
5.413.3.43	SSL_pending_d	2058
5.413.3.43	SSL_read_d	2058
5.413.3.44	SSL_set_accept_state_d	2058
5.413.3.44	SSL_set_bio_d	2059
5.413.3.44	SSL_set_connect_state_d	2059
5.413.3.44	SSL_set_fd_d	2059
5.413.3.44	SSL_set_read_ahead_d	2059
5.413.3.44	SSL_shutdown_d	2059
5.413.3.44	SSL_version_str_d	2059
5.413.3.44	SSL_want_d	2059
5.413.3.44	SSL_write_d	2059
5.413.3.44	SSLay_version_d	2059
5.413.3.45	SSLv23_client_method_d	2059
5.413.3.45	SSLv23_server_method_d	2059
5.413.3.45	free_d	2059
5.413.3.45	hdbclose_d	2059
5.413.3.45	hdbdefrag_d	2059
5.413.3.45	hdbdel_d	2060
5.413.3.45	hdbdecode_d	2060
5.413.3.45	hdberrmsg_d	2060
5.413.3.45	hdbfsiz_d	2060
5.413.3.45	hdbget_d	2060
5.413.3.46	hdbnew_d	2060
5.413.3.46	hdbopen_d	2060
5.413.3.46	hdboptimize_d	2060

5.413.3.463	chdbout_d	2060
5.413.3.464	chdbpath_d	2060
5.413.3.465	chdbputasync_d	2060
5.413.3.466	chdbbrnum_d	2060
5.413.3.467	chdbsetdfunit_d	2061
5.413.3.468	chdbsetmutex_d	2061
5.413.3.469	chdbsync_d	2061
5.413.3.470	chdbtune_d	2061
5.413.3.471	cllistdel_d	2061
5.413.3.472	cllistnum_d	2061
5.413.3.473	cllistval_d	2061
5.413.3.474	cnbdbdel_d	2061
5.413.3.475	cnbdbdup_d	2061
5.413.3.476	cnbdbfwkeys_d	2061
5.413.3.477	cnbdbget3_d	2061
5.413.3.478	cnbdbget_d	2061
5.413.3.479	cnbdbgetboth_d	2061
5.413.3.480	cnbdbiterinit_d	2061
5.413.3.481	cnbdbiternext2_d	2062
5.413.3.482	cnbdbnew2_d	2062
5.413.3.483	cnbdbout_d	2062
5.413.3.484	cnbdbputkeep_d	2062
5.413.3.485	cnbdbbrnum_d	2062
5.413.3.486	ctreeclear_d	2062
5.413.3.487	ctreekeys_d	2062
5.413.3.488	ctreevals_d	2062
5.413.3.489	ctversion_d	2062
5.413.3.490	DLsv1_server_method_d	2062
5.413.3.491	uncompress_d	2062
5.413.3.492	uf8proc_category_d	2062
5.413.3.493	uf8proc_category_string_d	2062
5.413.3.494	uf8proc_errmsg_d	2062
5.413.3.495	uf8proc_get_property_d	2063
5.413.3.496	uf8proc_iterate_d	2063
5.413.3.497	uf8proc_version_d	2063
5.413.3.498	X509_check_host_d	2063
5.413.3.499	X509_check_issued_d	2063
5.413.3.500	X509_email_free_d	2063
5.413.3.501	X509_get1_ocsp_d	2063

5.413.3.50X509_get_ext_count_d	2063
5.413.3.50X509_get_ext_d	2063
5.413.3.50X509_get_subject_name_d	2063
5.413.3.50X509_LOOKUP_file_d	2063
5.413.3.50X509_NAME_ENTRY_get_data_d	2063
5.413.3.50X509_NAME_get_entry_d	2063
5.413.3.50X509_NAME_get_index_by_NID_d	2064
5.413.3.50X509_NAME_get_text_by_NID_d	2064
5.413.3.51X509_NAME_online_d	2064
5.413.3.51X509_STORE_add_lookup_d	2064
5.413.3.51X509_STORE_CTX_free_d	2064
5.413.3.51X509_STORE_CTX_get_current_cert_d	2064
5.413.3.51X509_STORE_CTX_get_error_d	2064
5.413.3.51X509_STORE_CTX_get_error_depth_d	2064
5.413.3.51X509_STORE_CTX_init_d	2064
5.413.3.51X509_STORE_CTX_new_d	2064
5.413.3.51X509_STORE_CTX_set_chain_d	2064
5.413.3.51X509_STORE_free_d	2064
5.413.3.52X509_STORE_load_locations_d	2065
5.413.3.52X509_STORE_new_d	2065
5.413.3.52X509_STORE_set_flags_d	2065
5.413.3.52X509_verify_cert_d	2065
5.413.3.52X509_verify_cert_error_string_d	2065
5.413.3.52X509mlAddSibling_d	2065
5.413.3.52X509mlBufferContent_d	2065
5.413.3.52X509mlBufferCreate_d	2065
5.413.3.52X509mlBufferFree_d	2065
5.413.3.52X509mlBufferLength_d	2065
5.413.3.53X509mlCleanupGlobals_d	2065
5.413.3.53X509mlCleanupParser_d	2065
5.413.3.53X509mlCtxtReadMemory_d	2066
5.413.3.53X509mlDocDumpFormatMemory_d	2066
5.413.3.53X509mlEncodeEntitiesReentrant_d	2066
5.413.3.53X509mlFreeDoc_d	2066
5.413.3.53X509mlFreeNode_d	2066
5.413.3.53X509mlFreeParserCtxt_d	2066
5.413.3.53X509mlInitParser_d	2066
5.413.3.53X509mlMemoryDump_d	2066
5.413.3.54X509mlNewNode_d	2066

5.413.3.541	xmlNewParserCtxt_d	2066
5.413.3.542	xmlNodeBufGetContent_d	2066
5.413.3.543	xmlNodeSetContent_d	2066
5.413.3.544	xmlParserVersion_d	2067
5.413.3.545	xmlSetProp_d	2067
5.413.3.546	xmlXPathEvalExpression_d	2067
5.413.3.547	xmlXPathFreeContext_d	2067
5.413.3.548	xmlXPathFreeObject_d	2067
5.413.3.549	xmlXPathNewContext_d	2067
5.413.3.550	xmlXPathRegisterNs_d	2067
5.413.3.551	xmlVersion_d	2067
5.414	src/queries.h File Reference	2068
5.414.1	Define Documentation	2071
5.414.1.1	AUTH_GET_BY_ADDRESS	2071
5.414.1.2	AUTH_GET_BY_USERID	2071
5.414.1.3	AUTH_UPDATE_LEGACY_TO_STACIE	2071
5.414.1.4	AUTH_UPDATE_USER_LOCK	2071
5.414.1.5	DELETE_CONTACT	2071
5.414.1.6	DELETE_CONTACT_DETAILS	2071
5.414.1.7	DELETE_FOLDER	2071
5.414.1.8	DELETE_MESSAGE	2072
5.414.1.9	DELETE_MESSAGE_TAG	2072
5.414.1.10	DELETE_MESSAGE_TAGS	2072
5.414.1.11	DELETE_OBJECT	2072
5.414.1.12	DELETE_SIGNATURE	2072
5.414.1.13	DELETE_USER	2072
5.414.1.14	DELETE_USER_CONFIG	2072
5.414.1.15	FETCH_SIGNATURE	2072
5.414.1.16	INSERT_CONTACT	2072
5.414.1.17	INSERT_FOLDER	2072
5.414.1.18	INSERT_MESSAGE	2072
5.414.1.19	INSERT_MESSAGE_DUPLICATE	2073
5.414.1.20	INSERT_MESSAGE_TAG	2073
5.414.1.21	INSERT_OBJECT	2073
5.414.1.22	INSERT_RECEIVING	2073
5.414.1.23	INSERT_SIGNATURE	2073
5.414.1.24	INSERT_TRANSMITTING	2073
5.414.1.25	META_FETCH_MAIL_KEYS	2073
5.414.1.26	META_FETCH_SHARD	2073

5.414.1.27	META_FETCH_USER	2073
5.414.1.28	META_INSERT_MAIL_KEYS	2073
5.414.1.29	META_INSERT_SHARD	2073
5.414.1.30	QUERIES_INIT	2074
5.414.1.31	REGISTER_CHECK_USERNAME	2074
5.414.1.32	REGISTER_FETCH_BLOCKLIST	2074
5.414.1.33	REGISTER_INSERT_DISPATCH	2074
5.414.1.34	REGISTER_INSERT_FOLDER_NAME	2074
5.414.1.35	REGISTER_INSERT_LOG	2074
5.414.1.36	REGISTER_INSERT_MAILBOXES	2074
5.414.1.37	REGISTER_INSERT_PROFILE	2074
5.414.1.38	REGISTER_INSERT_STACIE_REALMS	2074
5.414.1.39	REGISTER_INSERT_STACIE_USER	2075
5.414.1.40	RENAME_FOLDER	2075
5.414.1.41	SELECT_AGENTS	2075
5.414.1.42	SELECT_ALERTS	2075
5.414.1.43	SELECT_ALL_MESSAGE_TAGS	2075
5.414.1.44	SELECT_AUTOREPLY	2075
5.414.1.45	SELECT_CONFIG	2075
5.414.1.46	SELECT_CONTACT_DETAILS	2075
5.414.1.47	SELECT_CONTACTS	2075
5.414.1.48	SELECT_DOMAINS	2075
5.414.1.49	SELECT_FILTERS	2076
5.414.1.50	SELECT_FOLDERS	2076
5.414.1.51	SELECT_HOST_NUMBER	2076
5.414.1.52	SELECT_MAILBOX_ADDRESS	2076
5.414.1.53	SELECT_MAILBOX_ADDRESS_ANY	2076
5.414.1.54	SELECT_MAILBOX_ALIASES	2076
5.414.1.55	SELECT_MESSAGE_FOLDER	2076
5.414.1.56	SELECT_MESSAGE_TAGS	2076
5.414.1.57	SELECT_MESSAGES	2076
5.414.1.58	SELECT_MESSAGES_ROLLOUT	2077
5.414.1.59	SELECT_PATTERNS	2077
5.414.1.60	SELECT_PREFS_INBOUND	2077
5.414.1.61	SELECT_RECEIVING	2077
5.414.1.62	SELECT_TRANSMITTING	2077
5.414.1.63	SELECT_USER	2077
5.414.1.64	SELECT_USER_AUTH	2077
5.414.1.65	SELECT_USER_CONFIG	2077

5.414.1.66	SELECT_USER_RECORD	2078
5.414.1.67	SELECT_USER_SALT	2078
5.414.1.68	SELECT_USER_STORAGE_KEYS	2078
5.414.1.69	SELECT_USERNUM_AUTH	2078
5.414.1.70	SELECT_USERNUM_AUTH_LEGACY	2078
5.414.1.71	SELECT_USERS_AUTH	2078
5.414.1.72	SMTP_SELECT_USER_AUTH	2078
5.414.1.73	STATISTICS_GET_EMAILS_RECEIVED_TODAY	2078
5.414.1.74	STATISTICS_GET_EMAILS_RECEIVED_WEEK	2078
5.414.1.75	STATISTICS_GET_EMAILS_SENT_TODAY	2078
5.414.1.76	STATISTICS_GET_EMAILS_SENT_WEEK	2079
5.414.1.77	STATISTICS_GET_TOTAL_USERS	2079
5.414.1.78	STATISTICS_GET_USERS_CHECKED_EMAIL_TODAY	2079
5.414.1.79	STATISTICS_GET_USERS_CHECKED_EMAIL_WEEK	2079
5.414.1.80	STATISTICS_GET_USERS_REGISTERED_TODAY	2079
5.414.1.81	STATISTICS_GET_USERS_REGISTERED_WEEK	2079
5.414.1.82	STATISTICS_GET_USERS_SENT_EMAIL_TODAY	2079
5.414.1.83	STATISTICS_GET_USERS_SENT_EMAIL_WEEK	2079
5.414.1.84	STMTS_INIT	2079
5.414.1.85	UPDATE_ALERTS_ACKNOWLEDGE	2079
5.414.1.86	UPDATE_CONTACT	2080
5.414.1.87	UPDATE_CONTACT_STAMP	2080
5.414.1.88	UPDATE_FOLDER	2080
5.414.1.89	UPDATE_LOG_IMAP	2080
5.414.1.90	UPDATE_LOG_POP	2080
5.414.1.91	UPDATE_LOG_RECEIVED	2080
5.414.1.92	UPDATE_LOG_SENT	2080
5.414.1.93	UPDATE_LOG_WEB	2080
5.414.1.94	UPDATE_MESSAGE_FLAGS_ADD	2080
5.414.1.95	UPDATE_MESSAGE_FLAGS_REMOVE	2080
5.414.1.96	UPDATE_MESSAGE_FLAGS_REPLACE	2081
5.414.1.97	UPDATE_MESSAGE_FOLDER	2081
5.414.1.98	UPDATE_MESSAGE_VISIBILITY	2081
5.414.1.99	UPDATE_SIGNATURE_FLAGS_ADD	2081
5.414.1.100	UPDATE_SIGNATURE_FLAGS_REMOVE	2081
5.414.1.101	UPDATE_USER_QUOTA_ADD	2081
5.414.1.102	UPDATE_USER_QUOTA_SUBTRACT	2081
5.414.1.103	UPDATE_USER_STORAGE_KEYS	2081
5.414.1.104	UPSERT_CONTACT_DETAIL	2081

5.414.1.105PSERT_USER_CONFIG	2081
5.414.2 Variable Documentation	2082
5.414.2.1 common	2082
5.414.2.2 queries	2082
5.414.2.3 STMTS_INIT	2082
5.415src/servers/dmtp/parse.c File Reference	2083
5.416src/servers/http/parse.c File Reference	2084
5.416.1 Function Documentation	2084
5.416.1.1 get_header_opt	2084
5.416.1.2 get_header_value_noopt	2085
5.416.1.3 http_parse_context	2085
5.416.1.4 http_parse_header	2085
5.416.1.5 http_parse_method	2086
5.416.1.6 http_parse_origin	2086
5.416.1.7 http_parse_pairs	2087
5.416.1.8 multipart_get_boundary	2087
5.417src/servers/imap/parse.c File Reference	2088
5.417.1 Function Documentation	2088
5.417.1.1 imap_command_log_safe	2088
5.417.1.2 imap_command_parser	2089
5.417.1.3 imap_get_ar_ar	2089
5.417.1.4 imap_get_ptr	2089
5.417.1.5 imap_get_st_ar	2090
5.417.1.6 imap_get_type_ar	2090
5.417.1.7 imap_parse_arguments	2090
5.417.1.8 imap_parse_array	2091
5.417.1.9 imap_parse_astring	2091
5.417.1.10imap_parse_literal	2092
5.417.1.11imap_parse_nstring	2092
5.417.1.12imap_parse_qstring	2093
5.418src/servers/pop/parse.c File Reference	2094
5.418.1 Function Documentation	2094
5.418.1.1 pop_num_parse	2094
5.418.1.2 pop_pass_parse	2094
5.418.1.3 pop_top_parse	2095
5.418.1.4 pop_user_parse	2095
5.419src/servers/smtp/parse.c File Reference	2096
5.419.1 Function Documentation	2096
5.419.1.1 smtp_parse_auth	2096

5.419.1.2 smtp_parse_helo_domain	2096
5.419.1.3 smtp_parse_mail_from_path	2097
5.419.1.4 smtp_parse_rcpt_to	2097
5.420src/web/portal/parse.c File Reference	2098
5.420.1 Function Documentation	2098
5.420.1.1 portal_parse_action	2098
5.420.1.2 portal_parse_context	2098
5.420.1.3 portal_parse_json_str_array	2099
5.420.1.4 portal_parse_sections	2099
5.421src/servers/dmtp/session.c File Reference	2100
5.421.1 Function Documentation	2100
5.421.1.1 dmtp_session_destroy	2100
5.421.1.2 dmtp_session_reset	2100
5.422src/servers/smtp/session.c File Reference	2101
5.422.1 Function Documentation	2101
5.422.1.1 smtp_add_inbound	2101
5.422.1.2 smtp_add_outbound	2102
5.422.1.3 smtp_add_recipient	2102
5.422.1.4 smtp_check_duplicate_recipient	2102
5.422.1.5 smtp_free_inbound	2102
5.422.1.6 smtp_free_outbound	2103
5.422.1.7 smtp_free_recipients	2103
5.422.1.8 smtp_list_free_filter	2103
5.422.1.9 smtp_session_destroy	2104
5.422.1.10smtp_session_reset	2104
5.423src/servers/http/content.c File Reference	2105
5.423.1 Function Documentation	2106
5.423.1.1 http_content_load_directory	2106
5.423.1.2 http_content_load_fonts	2106
5.423.1.3 http_content_refresh	2106
5.423.1.4 http_content_start	2107
5.423.1.5 http_content_stop	2107
5.423.1.6 http_free_content	2107
5.423.1.7 http_get_static	2107
5.423.1.8 http_get_template	2108
5.423.1.9 http_load_file	2108
5.423.1.10http_page_free	2108
5.423.1.11http_page_get	2109
5.423.2 Variable Documentation	2109

5.423.2.1 content	2109
5.423.2.2 fonts	2109
5.423.2.3 pages	2109
5.423.2.4 templates	2109
5.424src/servers/http/http.c File Reference	2110
5.424.1 Function Documentation	2110
5.424.1.1 http_body	2110
5.424.1.2 http_close	2110
5.424.1.3 http_init	2111
5.424.1.4 http_process	2111
5.424.1.5 http_requeue	2111
5.425src/servers/http/response.c File Reference	2112
5.425.1 Function Documentation	2112
5.425.1.1 http_response	2112
5.425.1.2 http_response_allow_cross	2113
5.425.1.3 http_response_connection	2113
5.425.1.4 http_response_cookie	2113
5.425.1.5 http_response_header	2113
5.425.1.6 http_response_options	2114
5.425.1.7 http_response_status	2114
5.426src/servers/imap/fetch.c File Reference	2115
5.426.1 Function Documentation	2115
5.426.1.1 imap_duplicate_messages	2115
5.426.1.2 imap_fetch_body	2116
5.426.1.3 imap_fetch_body_header	2116
5.426.1.4 imap_fetch_body_mime	2116
5.426.1.5 imap_fetch_body_part	2116
5.426.1.6 imap_fetch_body_portion	2116
5.426.1.7 imap_fetch_body_tag	2116
5.426.1.8 imap_fetch_bodystructure	2117
5.426.1.9 imap_fetch_envelope	2117
5.426.1.10imap_fetch_free_items	2117
5.426.1.11imap_fetch_message	2117
5.426.1.12imap_fetch_parse_partial	2117
5.426.1.13imap_fetch_return_header	2117
5.426.1.14imap_fetch_return_message	2118
5.426.1.15imap_fetch_return_mime	2118
5.426.1.16imap_fetch_return_text	2118
5.426.1.17imap_narrow_messages	2118

5.426.1.18imap_parse_dataitems	2118
5.426.1.19imap_valid_sequence	2118
5.427src/servers/imap/fetch_response.c File Reference	2120
5.427.1 Function Documentation	2120
5.427.1.1 imap_fetch_response_add	2120
5.427.1.2 imap_fetch_response_free	2120
5.428src/servers/imap/flags.c File Reference	2121
5.428.1 Function Documentation	2121
5.428.1.1 imap_flag_action	2121
5.428.1.2 imap_flag_parse	2121
5.428.1.3 imap_get_flag	2121
5.428.1.4 imap_update_flags	2121
5.429src/web/portal/flags.c File Reference	2122
5.429.1 Function Documentation	2122
5.429.1.1 portal_message_flags_array	2122
5.429.1.2 portal_message_tags_array	2122
5.429.1.3 portal_parse_flags	2122
5.430src/servers/imap/imap.c File Reference	2123
5.430.1 Function Documentation	2124
5.430.1.1 imap_append	2124
5.430.1.2 imap_capability	2124
5.430.1.3 imap_check	2124
5.430.1.4 imap_close	2124
5.430.1.5 imap_copy	2124
5.430.1.6 imap_create	2125
5.430.1.7 imap_delete	2125
5.430.1.8 imap_examine	2125
5.430.1.9 imap_expunge	2125
5.430.1.10imap_fetch	2126
5.430.1.11imap_id	2126
5.430.1.12imap_idle	2126
5.430.1.13imap_init	2126
5.430.1.14imap_invalid	2126
5.430.1.15imap_list	2127
5.430.1.16imap_login	2127
5.430.1.17imap_logout	2127
5.430.1.18imap_lsub	2127
5.430.1.19imap_noop	2128
5.430.1.20imap_rename	2128

5.430.1.2	imap_search	2128
5.430.1.22	imap_select	2128
5.430.1.23	imap_starttls	2128
5.430.1.24	imap_status	2129
5.430.1.25	imap_store	2129
5.430.1.26	imap_subscribe	2129
5.430.1.27	imap_unsubscribe	2129
5.431	src/servers/imap/output.c File Reference	2130
5.431.1	Function Documentation	2130
5.431.1.1	imap_build_array	2130
5.431.1.2	imap_build_array_isliteral	2130
5.432	src/servers/imap/parse_address.c File Reference	2131
5.432.1	Function Documentation	2131
5.432.1.1	imap_parse_address	2131
5.432.1.2	imap_parse_address_breaker	2131
5.432.1.3	imap_parse_address_part	2131
5.432.1.4	imap_parse_address_put	2131
5.433	src/servers/imap/range.c File Reference	2132
5.433.1	Function Documentation	2132
5.433.1.1	imap_range_build	2132
5.434	src/servers/molten/molten.c File Reference	2133
5.434.1	Function Documentation	2133
5.434.1.1	molten_init	2133
5.434.1.2	molten_invalid	2133
5.434.1.3	molten_quit	2133
5.434.1.4	molten_stats	2133
5.435	src/servers/molten/molten.h File Reference	2134
5.435.1	Function Documentation	2134
5.435.1.1	molten_compare	2134
5.435.1.2	molten_init	2134
5.435.1.3	molten_invalid	2134
5.435.1.4	molten_parse	2134
5.435.1.5	molten_quit	2135
5.435.1.6	molten_session_destroy	2135
5.435.1.7	molten_sort	2135
5.435.1.8	molten_stats	2135
5.436	src/servers/pop/mailbox.c File Reference	2136
5.436.1	Function Documentation	2136
5.436.1.1	pop_get_last	2136

5.436.1.2 pop_get_message	2136
5.436.1.3 pop_total_messages	2137
5.436.1.4 pop_total_size	2137
5.437src/servers/pop/pop.c File Reference	2138
5.437.1 Function Documentation	2139
5.437.1.1 pop_capa	2139
5.437.1.2 pop_dele	2139
5.437.1.3 pop_init	2139
5.437.1.4 pop_invalid	2140
5.437.1.5 pop_last	2140
5.437.1.6 pop_list	2140
5.437.1.7 pop_noop	2141
5.437.1.8 pop_pass	2141
5.437.1.9 pop_quit	2141
5.437.1.10pop_retr	2142
5.437.1.11pop_rset	2142
5.437.1.12pop_starttls	2142
5.437.1.13pop_stat	2143
5.437.1.14pop_top	2143
5.437.1.15pop_uidl	2143
5.437.1.16pop_user	2144
5.438src/servers/smtp/accept.c File Reference	2145
5.438.1 Function Documentation	2145
5.438.1.1 smtp_accept_message	2145
5.438.1.2 smtp_rollout	2145
5.438.1.3 smtp_store_message	2146
5.438.1.4 smtp_store_spamsig	2146
5.439src/servers/smtp/checkers.c File Reference	2147
5.439.1 Function Documentation	2147
5.439.1.1 smtp_add_bypass_entry	2147
5.439.1.2 smtp_bypass_check	2147
5.439.1.3 smtp_check_filters	2148
5.439.1.4 smtp_check_greylist	2148
5.439.1.5 smtp_check_rbl	2148
5.440src/servers/smtp/smtp.c File Reference	2150
5.440.1 Function Documentation	2151
5.440.1.1 smtp_auth_login	2151
5.440.1.2 smtp_auth_plain	2151
5.440.1.3 smtp_data	2151

5.440.1.4 smtp_data_finish	2151
5.440.1.5 smtp_data_inbound	2151
5.440.1.6 smtp_data_outbound	2151
5.440.1.7 smtp_data_read	2152
5.440.1.8 smtp_disabled	2152
5.440.1.9 smtp_ehlo	2152
5.440.1.10smtp_helo	2152
5.440.1.11smtp_init	2153
5.440.1.12smtp_invalid	2153
5.440.1.13smtp_mail_from	2153
5.440.1.14smtp_noop	2154
5.440.1.15smtp_quit	2154
5.440.1.16smtp_rcpt_to	2154
5.440.1.17smtp_rset	2155
5.440.1.18smtp_starttls	2155
5.440.1.19submission_init	2155
5.441src/servers/smtp/transmit.c File Reference	2156
5.441.1 Function Documentation	2156
5.441.1.1 smtp_bounce	2156
5.441.1.2 smtp_forward_message	2156
5.441.1.3 smtp_relay_message	2156
5.441.1.4 smtp_reply	2157
5.441.1.5 smtp_send_message	2157
5.442src/web/contact/abuse.c File Reference	2158
5.442.1 Function Documentation	2158
5.442.1.1 contact_abuse_checks	2158
5.442.1.2 contact_abuse_increment_history	2158
5.443src/web/register/abuse.c File Reference	2159
5.443.1 Function Documentation	2159
5.443.1.1 register_abuse_check_blocklist	2159
5.443.1.2 register_abuse_checks	2159
5.443.1.3 register_abuse_increment_history	2160
5.443.1.4 register_blocklist_free	2160
5.443.1.5 register_blocklist_update	2160
5.443.2 Variable Documentation	2160
5.443.2.1 register_blocklist	2160
5.443.2.2 register_blocklist_lock	2161
5.444src/web/contact/business.c File Reference	2162
5.444.1 Function Documentation	2162

5.444.1.1	contact_business	2162
5.444.1.2	contact_business_add_error	2162
5.444.1.3	contact_business_valid_email	2163
5.445	src/web/register/business.c File Reference	2164
5.445.1	Function Documentation	2164
5.445.1.1	register_business_step1	2164
5.445.1.2	register_business_step2	2164
5.445.1.3	register_business_validate_password	2165
5.445.1.4	register_business_validate_username	2165
5.446	src/web/contact/contact.c File Reference	2166
5.446.1	Function Documentation	2166
5.446.1.1	contact_print_form	2166
5.446.1.2	contact_print_message	2166
5.446.1.3	contact_process	2167
5.447	src/web/contact/contact.h File Reference	2168
5.447.1	Function Documentation	2168
5.447.1.1	contact_abuse_checks	2168
5.447.1.2	contact_abuse_increment_history	2169
5.447.1.3	contact_business	2169
5.447.1.4	contact_business_add_error	2169
5.447.1.5	contact_business_valid_email	2170
5.447.1.6	contact_print_form	2170
5.447.1.7	contact_print_message	2170
5.447.1.8	contact_process	2171
5.448	src/web/json_api/endpoints.c File Reference	2172
5.448.1	Function Documentation	2172
5.448.1.1	api_endpoint_auth	2172
5.448.1.2	api_endpoint_change_password	2172
5.448.1.3	api_endpoint_delete_user	2172
5.448.1.4	api_endpoint_register	2172
5.449	src/web/json_api/helpers.c File Reference	2173
5.449.1	Function Documentation	2173
5.449.1.1	api_error	2173
5.449.1.2	api_response	2173
5.449.1.3	jansson_flags	2174
5.450	src/web/json_api/json_api.c File Reference	2175
5.450.1	Function Documentation	2175
5.450.1.1	json_api_dispatch	2175
5.451	src/web/json_api/json_api.h File Reference	2176

5.451.1 Function Documentation	2176
5.451.1.1 api_endpoint_auth	2176
5.451.1.2 api_endpoint_change_password	2176
5.451.1.3 api_endpoint_delete_user	2176
5.451.1.4 api_endpoint_register	2176
5.451.1.5 api_error	2177
5.451.1.6 api_response	2177
5.451.1.7 json_api_dispatch	2177
5.452src/web/portal/endpoint.c File Reference	2179
5.452.1 Function Documentation	2181
5.452.1.1 portal_debug	2181
5.452.1.2 portal_endpoint	2181
5.452.1.3 portal_endpoint_ad	2182
5.452.1.4 portal_endpoint_alert_acknowledge	2182
5.452.1.5 portal_endpoint_alert_list	2182
5.452.1.6 portal_endpoint_aliases	2183
5.452.1.7 portal_endpoint_attachments_add	2183
5.452.1.8 portal_endpoint_attachments_progress	2183
5.452.1.9 portal_endpoint_attachments_remove	2184
5.452.1.10portal_endpoint_auth	2184
5.452.1.11portal_endpoint_compare	2184
5.452.1.12portal_endpoint_config_edit	2185
5.452.1.13portal_endpoint_config_load	2185
5.452.1.14portal_endpoint_contacts_add	2185
5.452.1.15portal_endpoint_contacts_copy	2186
5.452.1.16portal_endpoint_contacts_edit	2186
5.452.1.17portal_endpoint_contacts_list	2186
5.452.1.18portal_endpoint_contacts_load	2187
5.452.1.19portal_endpoint_contacts_move	2187
5.452.1.20portal_endpoint_contacts_remove	2187
5.452.1.21portal_endpoint_cookies	2188
5.452.1.22portal_endpoint_error	2188
5.452.1.23portal_endpoint_folders_add	2189
5.452.1.24portal_endpoint_folders_list	2189
5.452.1.25portal_endpoint_folders_remove	2189
5.452.1.26portal_endpoint_folders_rename	2190
5.452.1.27portal_endpoint_folders_tags	2190
5.452.1.28portal_endpoint_logout	2190
5.452.1.29portal_endpoint_messages_compose	2191

5.452.1.30	portal_endpoint_messages_copy	2191
5.452.1.31	portal_endpoint_messages_flag	2191
5.452.1.32	portal_endpoint_messages_list	2192
5.452.1.33	portal_endpoint_messages_load	2192
5.452.1.34	portal_endpoint_messages_move	2192
5.452.1.35	portal_endpoint_messages_remove	2192
5.452.1.36	portal_endpoint_messages_send	2193
5.452.1.37	portal_endpoint_messages_tag	2193
5.452.1.38	portal_endpoint_messages_tags	2194
5.452.1.39	portal_endpoint_response	2194
5.452.1.40	portal_endpoint_scrape	2195
5.452.1.41	portal_endpoint_scrape_add	2195
5.452.1.42	portal_endpoint_search	2195
5.452.1.43	portal_endpoint_sort	2196
5.452.1.44	portal_get_upload_attachment	2196
5.452.1.45	portal_meta	2196
5.452.1.46	portal_settings_changepass	2196
5.452.1.47	portal_settings_identity	2197
5.452.1.48	portal_upload	2197
5.452.1.49	portal_validate_request	2197
5.453	src/web/portal/mail.c File Reference	2199
5.453.1	Function Documentation	2199
5.453.1.1	portal_outbound_checks	2199
5.453.1.2	portal_smtp_create_data	2200
5.453.1.3	portal_smtp_merge_headers	2200
5.453.1.4	portal_smtp_relay_message	2200
5.454	src/web/portal/methods.h File Reference	2202
5.454.1	Variable Documentation	2202
5.454.1.1	portal_methods	2202
5.455	src/web/portal/portal.c File Reference	2203
5.455.1	Function Documentation	2203
5.455.1.1	portal_print_login	2203
5.455.1.2	portal_process	2203
5.456	src/web/portal/portal.h File Reference	2205
5.456.1	Define Documentation	2209
5.456.1.1	MAGMA_PORTAL_VERSION	2209
5.456.2	Enumeration Type Documentation	2210
5.456.2.1	"@86	2210
5.456.2.2	"@87	2210

5.456.2.3 "@88	2210
5.456.2.4 "@89	2210
5.456.2.5 "@90	2211
5.456.2.6 "@91	2211
5.456.3 Function Documentation	2212
5.456.3.1 portal_config_collection	2212
5.456.3.2 portal_config_entry	2212
5.456.3.3 portal_config_entry_flags	2213
5.456.3.4 portal_contact_detail_flags	2213
5.456.3.5 portal_contact_details	2213
5.456.3.6 portal_debug	2213
5.456.3.7 portal_endpoint	2214
5.456.3.8 portal_endpoint_ad	2214
5.456.3.9 portal_endpoint_alert_acknowledge	2214
5.456.3.10portal_endpoint_alert_list	2215
5.456.3.11portal_endpoint_aliases	2215
5.456.3.12portal_endpoint_attachments_add	2215
5.456.3.13portal_endpoint_attachments_progress	2216
5.456.3.14portal_endpoint_attachments_remove	2216
5.456.3.15portal_endpoint_auth	2216
5.456.3.16portal_endpoint_compare	2216
5.456.3.17portal_endpoint_config_edit	2217
5.456.3.18portal_endpoint_config_load	2217
5.456.3.19portal_endpoint_contacts_add	2218
5.456.3.20portal_endpoint_contacts_copy	2218
5.456.3.21portal_endpoint_contacts_edit	2218
5.456.3.22portal_endpoint_contacts_list	2218
5.456.3.23portal_endpoint_contacts_load	2219
5.456.3.24portal_endpoint_contacts_move	2219
5.456.3.25portal_endpoint_contacts_remove	2220
5.456.3.26portal_endpoint_cookies	2220
5.456.3.27portal_endpoint_error	2220
5.456.3.28portal_endpoint_folders_add	2221
5.456.3.29portal_endpoint_folders_list	2221
5.456.3.30portal_endpoint_folders_remove	2222
5.456.3.31portal_endpoint_folders_rename	2222
5.456.3.32portal_endpoint_folders_tags	2222
5.456.3.33portal_endpoint_logout	2222
5.456.3.34portal_endpoint_messages_compose	2223

5.456.3.35portal_endpoint_messages_copy	2223
5.456.3.36portal_endpoint_messages_flag	2223
5.456.3.37portal_endpoint_messages_list	2224
5.456.3.38portal_endpoint_messages_load	2224
5.456.3.39portal_endpoint_messages_move	2224
5.456.3.40portal_endpoint_messages_remove	2225
5.456.3.41portal_endpoint_messages_send	2225
5.456.3.42portal_endpoint_messages_tag	2225
5.456.3.43portal_endpoint_messages_tags	2226
5.456.3.44portal_endpoint_response	2226
5.456.3.45portal_endpoint_scrape	2227
5.456.3.46portal_endpoint_scrape_add	2227
5.456.3.47portal_endpoint_search	2228
5.456.3.48portal_endpoint_sort	2228
5.456.3.49portal_folder_contacts_add	2228
5.456.3.50portal_folder_contacts_remove	2228
5.456.3.51portal_folder_mail_add	2229
5.456.3.52portal_folder_mail_remove	2229
5.456.3.53portal_message_attachments	2230
5.456.3.54portal_message_body	2230
5.456.3.55portal_message_flags_array	2230
5.456.3.56portal_message_header	2230
5.456.3.57portal_message_info	2231
5.456.3.58portal_message_meta	2231
5.456.3.59portal_message_security	2231
5.456.3.60portal_message_server	2231
5.456.3.61portal_message_source	2232
5.456.3.62portal_message_tags_array	2232
5.456.3.63portal_meta	2232
5.456.3.64portal_outbound_checks	2232
5.456.3.65portal_parse_action	2233
5.456.3.66portal_parse_context	2233
5.456.3.67portal_parse_flags	2233
5.456.3.68portal_parse_json_str_array	2234
5.456.3.69portal_parse_sections	2234
5.456.3.70portal_print_login	2234
5.456.3.71portal_process	2235
5.456.3.72portal_settings_identity	2235
5.456.3.73portal_smtp_create_data	2235

5.456.3.74portal_smtp_merge_headers	2236
5.456.3.75portal_smtp_relay_message	2236
5.456.3.76portal_upload	2237
5.457src/web/register/captcha.c File Reference	2238
5.457.1 Function Documentation	2238
5.457.1.1 register_captcha_generate	2238
5.457.1.2 register_captcha_random_font	2238
5.457.1.3 register_captcha_write_noise	2239
5.458src/web/register/register.c File Reference	2240
5.458.1 Function Documentation	2240
5.458.1.1 register_print_captcha	2240
5.458.1.2 register_print_message	2240
5.458.1.3 register_print_step1	2241
5.458.1.4 register_print_step2	2241
5.458.1.5 register_print_step3	2242
5.458.1.6 register_process	2242
5.459src/web/register/register.h File Reference	2243
5.459.1 Define Documentation	2244
5.459.1.1 REGISTER_PASSWORD_MAX_LENGTH	2244
5.459.1.2 REGISTER_PASSWORD_MIN_LENGTH	2244
5.459.1.3 REGISTER_USERNAME_MAX_LENGTH	2245
5.459.1.4 REGISTER_USERNAME_MIN_LENGTH	2245
5.459.2 Function Documentation	2245
5.459.2.1 register_abuse_check_blocklist	2245
5.459.2.2 register_abuse_checks	2245
5.459.2.3 register_abuse_increment_history	2245
5.459.2.4 register_blocklist_free	2246
5.459.2.5 register_blocklist_update	2246
5.459.2.6 register_business_step1	2246
5.459.2.7 register_business_step2	2247
5.459.2.8 register_business_validate_password	2247
5.459.2.9 register_business_validate_username	2247
5.459.2.10register_captcha_generate	2248
5.459.2.11register_captcha_random_font	2248
5.459.2.12register_captcha_write_noise	2248
5.459.2.13register_data_check_username	2249
5.459.2.14register_data_fetch_blocklist	2249
5.459.2.15register_data_insert_user	2249
5.459.2.16register_print_captcha	2250

5.459.2.17	register_print_message	2250
5.459.2.18	register_print_step1	2250
5.459.2.19	register_print_step2	2251
5.459.2.20	register_print_step3	2251
5.459.2.21	register_process	2252
5.459.2.22	register_session_cache	2252
5.459.2.23	register_session_free	2252
5.459.2.24	register_session_generate	2253
5.459.2.25	register_session_get	2253
5.460	src/web/statistics/statistics.h File Reference	2254
5.460.1	Enumeration Type Documentation	2254
5.460.1.1	"@92	2254
5.460.2	Function Documentation	2255
5.460.2.1	statistics_init	2255
5.460.2.2	statistics_process	2255
5.460.2.3	statistics_refresh	2255
5.461	src/web/teacher/teacher.c File Reference	2256
5.461.1	Function Documentation	2256
5.461.1.1	teacher_add_cookie	2256
5.461.1.2	teacher_add_error	2256
5.461.1.3	teacher_print_form	2257
5.461.1.4	teacher_print_message	2257
5.461.1.5	teacher_process	2257
5.462	src/web/teacher/teacher.h File Reference	2259
5.462.1	Function Documentation	2259
5.462.1.1	teacher_add_cookie	2259
5.462.1.2	teacher_add_error	2260
5.462.1.3	teacher_data_delete	2260
5.462.1.4	teacher_data_fetch	2260
5.462.1.5	teacher_data_free	2261
5.462.1.6	teacher_data_get	2261
5.462.1.7	teacher_data_save	2261
5.462.1.8	teacher_print_form	2262
5.462.1.9	teacher_print_message	2262
5.462.1.10	teacher_process	2262
5.463	src/web/web.h File Reference	2264

Chapter 1

Removing Legacy Authentication Support

- Database Remove the legacy field from the Users table.
- Config Remove the "salt" parameter from the global config.
- Queries Remove/update queries which reference the legacy field.
- Auth Remove the legacy struct from the `auth_t` typedef.
- Code Remove the legacy routines and any code which references the `auth->legacy` structure, and all references to the `auth_legacy_t` structure.
- Structures `auth_t` (legacy) `auth_legacy_t`

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

__attribute__	17
api_lookup_t	21
auth_legacy_t	22
auth_stacie_t	23
auth_t	25
batch_heap_t	28
cache_keys_t	29
cache_t	30
cached_object	31
cached_store_t	34
derror_t	36
dime_ctx	37
dime_record_t	38
dmime_chunk_key_t	41
dmime_common_headers_t	43
dmime_envelope_object_t	44
dmime_header_key_t	45
dmime_message_t	46
dmime_object_t	48
dmtpl_argument_t	51
dmtpl_command_key_t	52
dmtpl_command_t	53
dmtpl_session_t	54
dnskey	56
ds	59
ED25519_KEY	61
encrypt_ctx	62
encrypt_keypair_t	63
err_desc_t	64
ge25519_niels_t	65
ge25519_p1p1_t	66
ge25519_pniels_t	67
ge25519_t	68
ge_cached	69
ge_p1p1	70
ge_p2	71

ge_p3	72
ge_precomp	73
generated_data_t	74
http_content_t	75
http_data_t	76
http_page_t	77
imap_fetch_dataitems_t	78
imap_fetch_response_t	81
imap_folder_status_t	82
ip_t	83
key_pair_t	84
log_level_t	85
magma_keys_t	86
magma_t	88
mail_cache_t	103
mail_message_t	104
mail_mime_t	106
mappings_t	108
media_type_t	109
meta_stats_tag_t	110
multi_t	111
mx_record_t	114
nvp_t	115
object_cache_t	116
object_chunk	117
packedelem32_t	118
packedelem64_t	119
packedelem8_t	120
queue_t	121
random_data_t	122
register_session_t	123
relay_keys_t	125
relay_t	126
server_config_t	127
server_keys_t	129
server_t	130
sha_databuf_t	133
signet_field_key_t	134
signet_field_t	136
signet_t	138
smtp_inbound_filter_t	139
smtp_inbound_prefs_t	141
smtp_message_t	147
smtp_outbound_prefs_t	149
smtp_recipients_t	151
smtp_session_t	152
statistics_vp_t	155
subnet_t	156
teacher_data_t	157
test_data_t	159
tok_state_t	160

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

src/magma.c	821
src/magma.h	822
src/queries.h	2068
src/core/core.h	223
src/core/type.c	610
src/core/buckets/arrays.c	161
src/core/buckets/buckets.h	166
src/core/buckets/pool.c	181
src/core/buckets/stacked.c	187
src/core/checksum/adler.c	189
src/core/checksum/checksum.h	190
src/core/checksum/crc.c	194
src/core/checksum/fletcher.c	199
src/core/checksum/murmur.c	200
src/core/classify/ascii.c	201
src/core/classify/classify.h	205
src/core/compare/compare.h	209
src/core/compare/ends.c	214
src/core/compare/equal.c	215
src/core/compare/search.c	217
src/core/compare/starts.c	222
src/core/encodings/base64.c	232
src/core/encodings/encodings.h	237
src/core/encodings/hex.c	249
src/core/encodings/mappings.c	253
src/core/encodings/qp.c	254
src/core/encodings/url.c	255
src/core/encodings/zbase32.c	257
src/core/host/backtrace.c (A simple function for dumping a stack trace to standard output)	258
src/core/host/color.c	259
src/core/host/errors.c (Functions to help handle errno)	265
src/core/host/files.c	274
src/core/host/folder.c	277
src/core/host/host.c	278
src/core/host/host.h	279
src/core/host/ip.c (Generic interface for parsing, and manipulating the IP address structure)	307

src/core/host/ process.c	314
src/core/host/ signals.c (Functions to help handle signals)	318
src/core/host/ spool.c	319
src/core/host/ tcp.c (Generic fuctions for interaction with TCP/IP socket connections)	323
src/core/indexes/ cursors.c	326
src/core/indexes/ hashed.c	330
src/core/indexes/ indexes.h	336
src/core/indexes/ inx.c	348
src/core/indexes/ linked.c	355
src/core/memory/ align.c	364
src/core/memory/ bitwise.c	365
src/core/memory/ memory.c	370
src/core/memory/ memory.h	376
src/core/memory/ secure.c	386
src/core/parsers/ bitwise.c	367
src/core/parsers/ case.c	392
src/core/parsers/ line.c	398
src/core/parsers/ parsers.h	442
src/core/parsers/ time.c	473
src/core/parsers/ token.c	475
src/core/parsers/ trim.c	482
src/core/parsers/formats/ formats.h	394
src/core/parsers/formats/ nvp.c	396
src/core/parsers/numbers/ clamp.c	400
src/core/parsers/numbers/ digits.c	404
src/core/parsers/numbers/ numbers.c	407
src/core/parsers/numbers/ numbers.h	422
src/core/parsers/special/ bracket.c	471
src/core/parsers/special/ special.h	472
src/core/strings/ allocation.c	484
src/core/strings/ data.c	493
src/core/strings/ info.c	503
src/core/strings/ length.c	506
src/core/strings/ multi.c	510
src/core/strings/ nuller.c	516
src/core/strings/ opts.c	521
src/core/strings/ print.c	523
src/core/strings/ replace.c	527
src/core/strings/ shortcuts.c	528
src/core/strings/ strings.h	533
src/core/strings/ validate.c	579
src/core/thread/ keys.c	583
src/core/thread/ mutex.c	587
src/core/thread/ rwlock.c	590
src/core/thread/ thread.c	594
src/core/thread/ thread.h	599
src/engine/ engine.h	787
src/engine/config/ config.h	672
src/engine/config/cache/ cache.c	611
src/engine/config/cache/ cache.h	644
src/engine/config/cache/ keys.h	663
src/engine/config/global/ datatier.c	682
src/engine/config/global/ global.c	722
src/engine/config/global/ global.h	729
src/engine/config/global/ keys.h	664
src/engine/config/relay/ keys.h	665
src/engine/config/relay/ relay.c	735

src/engine/config/relay/relay.h	742
src/engine/config/servers/keys.h	666
src/engine/config/servers/servers.c	746
src/engine/config/servers/servers.h	752
src/engine/context/args.c	760
src/engine/context/context.h	762
src/engine/context/process.c	316
src/engine/context/sanity.c	770
src/engine/context/signal.c	771
src/engine/context/system.c	774
src/engine/context/thread.c	598
src/engine/controller/controller.h	777
src/engine/controller/protocol.c	781
src/engine/controller/queue.c	783
src/engine/log/log.c	788
src/engine/log/log.h	791
src/engine/status/build.c	795
src/engine/status/performance.c	797
src/engine/status/statistics.c	798
src/engine/status/status.c	807
src/engine/status/status.h	810
src/network/addresses.c	825
src/network/clients.c	829
src/network/connections.c	831
src/network/dmtp.h	835
src/network/http.h	850
src/network/imap.h	872
src/network/listeners.c	903
src/network/meta.h	905
src/network/network.h	931
src/network/options.c	952
src/network/pop.h	955
src/network/read.c	967
src/network/reverse.c	969
src/network/sessions.h	971
src/network/smtp.h	981
src/network/write.c	1009
src/objects/locks.c	1094
src/objects/objects.c	1157
src/objects/objects.h	1224
src/objects/serials.c	1217
src/objects/auth/auth.c	1013
src/objects/auth/auth.h	1016
src/objects/auth/datatier.c	683
src/objects/auth/legacy.c	1024
src/objects/auth/stacie.c	1025
src/objects/auth/username.c	1028
src/objects/config/config.c	1029
src/objects/config/config.h	677
src/objects/config/datatier.c	685
src/objects/contacts/contacts.c	1034
src/objects/contacts/contacts.h	1044
src/objects/contacts/datatier.c	687
src/objects/contacts/find.c	1055
src/objects/folders/contacts.c	1040
src/objects/folders/datatier.c	691
src/objects/folders/find.c	1057

src/objects/folders/folders.c	1059
src/objects/folders/folders.h	1073
src/objects/folders/messages.c	1084
src/objects/mail/cache.c	615
src/objects/mail/cleanup.c	1097
src/objects/mail/counters.c	1099
src/objects/mail/datatier.c	693
src/objects/mail/headers.c	1101
src/objects/mail/load_message.c	1109
src/objects/mail/mail.h	1111
src/objects/mail/mime.c	1145
src/objects/mail/objects.c	1155
src/objects/mail/parsing.c	1161
src/objects/mail/paths.c	1162
src/objects/mail/remove_message.c	1164
src/objects/mail/signatures.c	1165
src/objects/mail/store_message.c	1172
src/objects/messages/datatier.c	696
src/objects/messages/messages.c	1086
src/objects/messages/messages.h	1175
src/objects/messages/meta.c	1188
src/objects/meta/alerts.c	1194
src/objects/meta/alias.c	1195
src/objects/meta/crypto.c	1196
src/objects/meta/datatier.c	698
src/objects/meta/folders.c	1062
src/objects/meta/indexes.c	1209
src/objects/meta/locking.c	1210
src/objects/meta/meta.c	1192
src/objects/meta/meta.h	908
src/objects/meta/references.c	1212
src/objects/meta/serials.c	1215
src/objects/meta/updaters.c	1220
src/objects/sessions/sessions.c	1229
src/objects/sessions/sessions.h	973
src/objects/warehouse/datatier.c	707
src/objects/warehouse/domains.c	1244
src/objects/warehouse/patterns.c	1248
src/objects/warehouse/warehouse.c	1250
src/objects/warehouse/warehouse.h	1252
src/providers/providers.h	1938
src/providers/symbols.c	1977
src/providers/symbols.h	2010
src/providers/checkers/allocations.h	1259
src/providers/checkers/checkers.h	1260
src/providers/checkers/clamav.c	1271
src/providers/checkers/dkim.c	1276
src/providers/checkers/dspam.c	1280
src/providers/checkers/spf.c	1283
src/providers/compress/bzip.c	1286
src/providers/compress/compress.c	1288
src/providers/compress/compress.h	1292
src/providers/compress/engine.c	1302
src/providers/compress/lzo.c	1303
src/providers/compress/zlib.c	1305
src/providers/consumers/cache.c	618
src/providers/consumers/consumers.h	1307

src/providers/consumers/counters.c	1100
src/providers/consumers/deserialization.c	1321
src/providers/consumers/serialization.c	1326
src/providers/cryptography/ciphers.c	1331
src/providers/cryptography/cryptex.c	1334
src/providers/cryptography/cryptography.h	1342
src/providers/cryptography/digest.c	1389
src/providers/cryptography/ecies.c	1390
src/providers/cryptography/hash.c	1402
src/providers/cryptography/hmac.c	1404
src/providers/cryptography/openssl.c	1408
src/providers/cryptography/parameters.c	1414
src/providers/cryptography/random.c	1417
src/providers/cryptography/scramble.c	1422
src/providers/cryptography/symmetric.c	1432
src/providers/cryptography/tls.c	1436
src/providers/database/database.h	1442
src/providers/database/mysql.c	1466
src/providers/database/query.c	1472
src/providers/database/results.c	1475
src/providers/database/stmts.c	1485
src/providers/database/transaction.c	1492
src/providers/deprecated/cryptex.c	1338
src/providers/deprecated/deprecated.h	1494
src/providers/deprecated/ecies.c	1396
src/providers/deprecated/hmac.c	1406
src/providers/deprecated/scramble.c	1427
src/providers/deprecated/symmetric.c	1434
src/providers/dime/dime_ctx.c	1564
src/providers/dime/dime_ctx.h	1566
src/providers/dime/error_codes.c	1690
src/providers/dime/error_codes.h	1691
src/providers/dime/common/crypto.c	1197
src/providers/dime/common/crypto_pub.c	1510
src/providers/dime/common/dcrypto.h	1514
src/providers/dime/common/error.c	1519
src/providers/dime/common/error.h	1524
src/providers/dime/common/misc.c	1536
src/providers/dime/common/misc.h	1550
src/providers/dime/common/misc_pub.c	1556
src/providers/dime/common/network.c	1561
src/providers/dime/common/network.h	951
src/providers/dime/common/network_pub.c	1563
src/providers/dime/dmessage/common.h	1569
src/providers/dime/dmessage/crypto.h	1580
src/providers/dime/dmessage/dmsg.c	1590
src/providers/dime/dmessage/parse.h	1599
src/providers/dime/dmessage/parser.c	1603
src/providers/dime/dmtp/commands.c	1606
src/providers/dime/dmtp/commands.h	1622
src/providers/dime/ed25519/curve25519-donna-32bit.h	1637
src/providers/dime/ed25519/curve25519-donna-64bit.h	1639
src/providers/dime/ed25519/curve25519-donna-helpers.h	1641
src/providers/dime/ed25519/curve25519-donna-sse2.h	1642
src/providers/dime/ed25519/ed25519-donna-32bit-sse2.h	1644
src/providers/dime/ed25519/ed25519-donna-32bit-tables.h	1645
src/providers/dime/ed25519/ed25519-donna-64bit-sse2.h	1646

src/providers/dime/ed25519/ed25519-donna-64bit-tables.h	1647
src/providers/dime/ed25519/ed25519-donna-64bit-x86-32bit.h	1648
src/providers/dime/ed25519/ed25519-donna-64bit-x86.h	1649
src/providers/dime/ed25519/ed25519-donna-basepoint-table.h	1650
src/providers/dime/ed25519/ed25519-donna-batchverify.h	1651
src/providers/dime/ed25519/ed25519-donna-impl-base.h	1653
src/providers/dime/ed25519/ed25519-donna-impl-sse2.h	1654
src/providers/dime/ed25519/ed25519-donna-portable-identify.h	1655
src/providers/dime/ed25519/ed25519-donna-portable.h	1656
src/providers/dime/ed25519/ed25519-donna.h	1657
src/providers/dime/ed25519/ed25519-hash-custom.h	1661
src/providers/dime/ed25519/ed25519-hash.h	1662
src/providers/dime/ed25519/ed25519-randombytes-custom.h	1663
src/providers/dime/ed25519/ed25519-randombytes.h	1664
src/providers/dime/ed25519/ed25519.c	1665
src/providers/dime/ed25519/ed25519.h	1669
src/providers/dime/ed25519/modm-donna-32bit.h	1683
src/providers/dime/ed25519/modm-donna-64bit.h	1684
src/providers/dime/ed25519/regression.h	1685
src/providers/dime/ed25519/test-internals.c	1686
src/providers/dime/ed25519/test-ticks.h	1687
src/providers/dime/ed25519/test.c	1688
src/providers/dime/ed25519/fuzz/curve25519-ref10.c	1671
src/providers/dime/ed25519/fuzz/curve25519-ref10.h	1674
src/providers/dime/ed25519/fuzz/ed25519-donna-sse2.c	1675
src/providers/dime/ed25519/fuzz/ed25519-donna.c	1676
src/providers/dime/ed25519/fuzz/ed25519-donna.h	1658
src/providers/dime/ed25519/fuzz/ed25519-ref10.c	1677
src/providers/dime/ed25519/fuzz/ed25519-ref10.h	1679
src/providers/dime/ed25519/fuzz/fuzz-curve25519.c	1680
src/providers/dime/ed25519/fuzz/fuzz-ed25519.c	1681
src/providers/dime/sds/sds.c	1693
src/providers/dime/sds/sds.h	1699
src/providers/dime/sds/sdsalloc.h	1706
src/providers/dime/sds/testhelp.h	1707
src/providers/dime/signet-resolver/cache.c	624
src/providers/dime/signet-resolver/cache.h	648
src/providers/dime/signet-resolver/cache_pub.c	1708
src/providers/dime/signet-resolver/dmtp.c	1711
src/providers/dime/signet-resolver/dmtp.h	836
src/providers/dime/signet-resolver/dmtp_pub.c	1727
src/providers/dime/signet-resolver/dns.c	1730
src/providers/dime/signet-resolver/dns.h	1746
src/providers/dime/signet-resolver/mrec.c	1760
src/providers/dime/signet-resolver/mrec.h	1764
src/providers/dime/signet-resolver/mrec_pub.c	1768
src/providers/dime/signet-resolver/signet-ssl.h	1769
src/providers/dime/signet-resolver/ssl.c	1775
src/providers/dime/signet-resolver/ssl_pub.c	1784
src/providers/dime/signet/common.h	1573
src/providers/dime/signet/general.c	1786
src/providers/dime/signet/keys.c	585
src/providers/dime/signet/keys.h	667
src/providers/dime/signet/signet.c	1788
src/providers/dime/signet/signet.h	1806
src/providers/dime/util/encoding.c	1825
src/providers/dime/util/encoding.h	1826

src/providers/dime/util/encrypt.c	1827
src/providers/dime/util/encrypt.h	1830
src/providers/images/freetype.c	1832
src/providers/images/gd.c	1833
src/providers/images/images.h	1834
src/providers/images/jpeg.c	1837
src/providers/images/png.c	1838
src/providers/parsers/json.c	1839
src/providers/parsers/parsers.h	456
src/providers/parsers/utf8.c	1840
src/providers/parsers/xml.c	1842
src/providers/prime/prime.c	1895
src/providers/prime/prime.h	1900
src/providers/prime/cryptography/aes.c	1855
src/providers/prime/cryptography/cryptography.h	1381
src/providers/prime/cryptography/ed25519.c	1666
src/providers/prime/cryptography/secp256k1.c	1859
src/providers/prime/keys/keys.h	669
src/providers/prime/keys/orgs.c	1863
src/providers/prime/keys/users.c	1867
src/providers/prime/messages/messages.c	1088
src/providers/prime/messages/messages.h	1186
src/providers/prime/messages/chunks/chunks.c	1871
src/providers/prime/messages/chunks/chunks.h	1873
src/providers/prime/messages/chunks/encrypted.c	1882
src/providers/prime/messages/chunks/ephemeral.c	1884
src/providers/prime/messages/chunks/keys.c	1886
src/providers/prime/messages/chunks/signature.c	1888
src/providers/prime/messages/chunks/slots.c	1890
src/providers/prime/messages/parts/parts.c	1893
src/providers/prime/messages/parts/parts.h (DESCRIPTIONxxxGOESxxxHERE)	1894
src/providers/prime/signets/orgs.c	1865
src/providers/prime/signets/requests.c	1913
src/providers/prime/signets/signets.h	1915
src/providers/prime/signets/users.c	1869
src/providers/prime/transposition/transposition.h	1937
src/providers/prime/transposition/armored/armored.h	1920
src/providers/prime/transposition/armored/pem.c	1921
src/providers/prime/transposition/binary/binary.h	1924
src/providers/prime/transposition/binary/fields.c	1932
src/providers/prime/transposition/binary/headers.c	1106
src/providers/prime/transposition/binary/objects.c	1159
src/providers/prime/transposition/binary/reader.c	1934
src/providers/prime/transposition/binary/unpack.c	1936
src/providers/stacie/creation.c (Create cryptographically strong random STACIE salt, nonce, and shard values)	1940
src/providers/stacie/crypto.c (Use a realm key to encrypt/decrypt data)	1207
src/providers/stacie/passwords.c (Process passwords into STACIE key values)	1942
src/providers/stacie/realms.c (Derive STACIE realm key and vector shard values)	1945
src/providers/stacie/stacie.h (These functions implement the Safely Turning Authentication Credentials Into Entropy (STACIE) standard. This standard defines how process passwords into encryption keys, and/or authentication tokens, derive realm specific encryption keys, and perform symmetric encryption using the realm keys. The inputs passed into these functions must be sanitized, and normalized to ensure a deterministic output)	1947
src/providers/stacie/tokens.c (Derive STACIE token values)	1956
src/providers/storage/data.c	499
src/providers/storage/storage.h	1958
src/providers/storage/tank.c	1966
src/providers/storage/tokyo.c	1971

src/providers/storage/tree.c	1972
src/servers/servers.h	759
src/servers/dmtp/commands.c	1614
src/servers/dmtp/commands.h	1631
src/servers/dmtp/dmtp.c	1722
src/servers/dmtp/dmtp.h	844
src/servers/dmtp/parse.c	2083
src/servers/dmtp/session.c	2100
src/servers/http/commands.h	1632
src/servers/http/content.c	2105
src/servers/http/data.c	500
src/servers/http/errors.c	270
src/servers/http/http.c	2110
src/servers/http/http.h	852
src/servers/http/parse.c	2084
src/servers/http/response.c	2112
src/servers/http/sessions.c	1238
src/servers/imap/commands.c	1616
src/servers/imap/commands.h	1633
src/servers/imap/fetch.c	2115
src/servers/imap/fetch_response.c	2120
src/servers/imap/flags.c	2121
src/servers/imap/folders.c	1065
src/servers/imap/imap.c	2123
src/servers/imap/imap.h	873
src/servers/imap/messages.c	1090
src/servers/imap/output.c	2130
src/servers/imap/parse.c	2088
src/servers/imap/parse_address.c	2131
src/servers/imap/range.c	2132
src/servers/imap/search.c	219
src/servers/imap/sessions.c	1239
src/servers/molten/commands.c	1618
src/servers/molten/commands.h	1634
src/servers/molten/molten.c	2133
src/servers/molten/molten.h	2134
src/servers/molten/sessions.c	1240
src/servers/pop/commands.c	1619
src/servers/pop/commands.h	1635
src/servers/pop/mailbox.c	2136
src/servers/pop/parse.c	2094
src/servers/pop/pop.c	2138
src/servers/pop/pop.h	956
src/servers/pop/sessions.c	1241
src/servers/smtp/accept.c	2145
src/servers/smtp/checkers.c	2147
src/servers/smtp/commands.c	1620
src/servers/smtp/commands.h	1636
src/servers/smtp/datatier.c	708
src/servers/smtp/parse.c	2096
src/servers/smtp/relay.c	739
src/servers/smtp/session.c	2101
src/servers/smtp/smtp.c	2150
src/servers/smtp/smtp.h	983
src/servers/smtp/transmit.c	2156
src/web/web.h	2264
src/web/contact/abuse.c	2158

src/web/contact/business.c	2162
src/web/contact/contact.c	2166
src/web/contact/contact.h	2168
src/web/json_api/endpoints.c	2172
src/web/json_api/helpers.c	2173
src/web/json_api/json_api.c	2175
src/web/json_api/json_api.h	2176
src/web/portal/config.c	1032
src/web/portal/contacts.c	1043
src/web/portal/endpoint.c	2179
src/web/portal/flags.c	2122
src/web/portal/folders.c	1070
src/web/portal/mail.c	2199
src/web/portal/messages.c	1091
src/web/portal/methods.h	2202
src/web/portal/parse.c	2098
src/web/portal/portal.c	2203
src/web/portal/portal.h	2205
src/web/register/abuse.c	2159
src/web/register/business.c	2164
src/web/register/captcha.c	2238
src/web/register/datatier.c	715
src/web/register/register.c	2240
src/web/register/register.h	2243
src/web/register/sessions.c	1242
src/web/statistics/datatier.c	717
src/web/statistics/statistics.c	806
src/web/statistics/statistics.h	2254
src/web/teacher/datatier.c	719
src/web/teacher/teacher.c	2256
src/web/teacher/teacher.h	2259

Chapter 4

Data Structure Documentation

4.1 `__attribute__` Struct Reference

```
#include <http.h>
```

Data Fields

- struct {
 - `int_t` cookie
 - `int_t` connection } response
- `session_t` * session
- `http_method_t` method
- `inx_t` * pairs
- `inx_t` * headers
- `int_t` mode
- `int_t` merged
- `int_t` port
- `stringer_t` * host
- `stringer_t` * location
- `stringer_t` * cookie
- `stringer_t` * agent
- `stringer_t` * body
- union {
 - struct {
 - `uint64_t` id
 - `json_t` * request
 - `json_t` * params } portal };
- `bool_t` expunge
- `meta_user_t` * user
- `int_t` session_state
- `stringer_t` * username
- `uint64_t` usernum

4.1.1 Detailed Description

Definition at line 53 of file http.h.

4.1.2 Field Documentation

4.1.2.1 union { ... }

4.1.2.2 stringer_t * __attribute__::agent

Definition at line 64 of file http.h.

Referenced by __attribute__().

4.1.2.3 stringer_t * __attribute__::body

Definition at line 64 of file http.h.

4.1.2.4 int_t __attribute__::connection

Definition at line 57 of file http.h.

4.1.2.5 stringer_t * __attribute__::cookie

Definition at line 64 of file http.h.

4.1.2.6 int_t __attribute__::cookie

Definition at line 56 of file http.h.

4.1.2.7 bool_t __attribute__::expunge

Definition at line 12 of file pop.h.

4.1.2.8 inx_t * __attribute__::headers

Definition at line 62 of file http.h.

4.1.2.9 stringer_t* __attribute__::host

Definition at line 64 of file http.h.

Referenced by __attribute__().

4.1.2.10 uint64_t __attribute__::id

Definition at line 68 of file http.h.

4.1.2.11 stringer_t * __attribute__::location

Definition at line 64 of file http.h.

4.1.2.12 `int_t __attribute__::merged`

Definition at line 63 of file `http.h`.

4.1.2.13 `http_method_t __attribute__::method`

Definition at line 61 of file `http.h`.

4.1.2.14 `int_t __attribute__::mode`

Definition at line 63 of file `http.h`.

4.1.2.15 `inx_t* __attribute__::pairs`

Definition at line 62 of file `http.h`.

4.1.2.16 `json_t * __attribute__::params`

Definition at line 69 of file `http.h`.

4.1.2.17 `int_t __attribute__::port`

Definition at line 63 of file `http.h`.

4.1.2.18 `struct { ... } __attribute__::portal`**4.1.2.19 `json_t* __attribute__::request`**

Definition at line 69 of file `http.h`.

Referenced by `__attribute__()`.

4.1.2.20 `struct { ... } __attribute__::response`**4.1.2.21 `session_t* __attribute__::session`**

Definition at line 60 of file `http.h`.

4.1.2.22 `int_t __attribute__::session_state`

Definition at line 14 of file `pop.h`.

Referenced by `__attribute__()`.

4.1.2.23 `meta_user_t* __attribute__::user`

Definition at line 13 of file `pop.h`.

Referenced by `__attribute__()`.

4.1.2.24 stringer_t* __attribute__::username

Definition at line 15 of file pop.h.

Referenced by __attribute__().

4.1.2.25 uint64_t __attribute__::usernum

Definition at line 16 of file pop.h.

Referenced by __attribute__().

4.2 api_lookup_t Struct Reference

Data Fields

- [chr_t](#) * [string](#)
- void(* [callback](#))([connection_t](#) *con)

4.2.1 Detailed Description

Definition at line 10 of file json_api.c.

4.2.2 Field Documentation

4.2.2.1 void(* api_lookup_t::callback)(connection_t *con)

Referenced by json_api_dispatch().

4.2.2.2 chr_t* api_lookup_t::string

Definition at line 11 of file json_api.c.

Referenced by json_api_dispatch().

4.3 auth_legacy_t Struct Reference

```
#include <auth.h>
```

Data Fields

- [stringer_t](#) * key
- [stringer_t](#) * token

4.3.1 Detailed Description

Definition at line 26 of file auth.h.

4.3.2 Field Documentation

4.3.2.1 [stringer_t](#)* auth_legacy_t::key

Definition at line 27 of file auth.h.

Referenced by [auth_legacy\(\)](#), [auth_legacy_free\(\)](#), and [auth_login\(\)](#).

4.3.2.2 [stringer_t](#)* auth_legacy_t::token

Definition at line 28 of file auth.h.

Referenced by [auth_legacy\(\)](#), [auth_legacy_free\(\)](#), and [auth_login\(\)](#).

4.4 auth_stacie_t Struct Reference

```
#include <auth.h>
```

Data Fields

- struct {
 stringer_t * master
 stringer_t * password
} keys
- struct {
 stringer_t * verification
 stringer_t * ephemeral
} tokens

4.4.1 Detailed Description

Definition at line 31 of file auth.h.

4.4.2 Field Documentation

4.4.2.1 stringer_t* auth_stacie_t::ephemeral

Definition at line 40 of file auth.h.

Referenced by auth_response(), auth_stacie(), and auth_stacie_free().

4.4.2.2 struct { ... } auth_stacie_t::keys

Referenced by auth_login(), auth_stacie(), auth_stacie_free(), and register_data_insert_user().

4.4.2.3 stringer_t* auth_stacie_t::master

Definition at line 34 of file auth.h.

Referenced by auth_login(), auth_stacie(), auth_stacie_free(), and register_data_insert_user().

4.4.2.4 stringer_t* auth_stacie_t::password

Definition at line 35 of file auth.h.

Referenced by auth_stacie(), and auth_stacie_free().

4.4.2.5 struct { ... } auth_stacie_t::tokens

Referenced by auth_login(), auth_response(), auth_stacie(), auth_stacie_free(), and register_data_insert_user().

4.4.2.6 `stringer_t* auth_stacie_t::verification`

Definition at line 39 of file `auth.h`.

Referenced by `auth_login()`, `auth_stacie()`, `auth_stacie_free()`, and `register_data_insert_user()`.

4.5 auth_t Struct Reference

```
#include <auth.h>
```

Data Fields

- uint64_t `usernum`
- `stringer_t` * `username`
- struct {
 - uint8_t `locked` } `status`
- struct {
 - uint64_t `bonus`
 - `stringer_t` * `salt`
 - `stringer_t` * `nonce` } `seasoning`
- struct {
 - `stringer_t` * `master` } `keys`
- struct {
 - `stringer_t` * `ephemeral`
 - `stringer_t` * `verification` } `tokens`
- struct {
 - `stringer_t` * `key`
 - `stringer_t` * `token` } `legacy`

4.5.1 Detailed Description

Definition at line 45 of file `auth.h`.

4.5.2 Field Documentation

4.5.2.1 uint64_t auth_t::bonus

Definition at line 55 of file `auth.h`.

Referenced by `auth_data_fetch()`, `auth_login()`, and `auth_response()`.

4.5.2.2 stringer_t* auth_t::ephemeral

Definition at line 65 of file `auth.h`.

Referenced by `auth_free()`, and `auth_response()`.

4.5.2.3 stringer_t* auth_t::key

Definition at line 70 of file auth.h.

Referenced by auth_free(), and auth_login().

4.5.2.4 struct { ... } auth_t::keys

Referenced by api_endpoint_auth(), auth_free(), auth_login(), imap_login(), pop_pass(), and portal_endpoint_auth().

4.5.2.5 struct { ... } auth_t::legacy

Referenced by auth_data_fetch(), auth_free(), and auth_login().

4.5.2.6 uint8_t auth_t::locked

Definition at line 51 of file auth.h.

Referenced by auth_data_fetch(), auth_login(), imap_login(), pop_pass(), portal_endpoint_auth(), smtp_auth_login(), and smtp_auth_plain().

4.5.2.7 stringer_t* auth_t::master

Definition at line 61 of file auth.h.

Referenced by api_endpoint_auth(), auth_free(), auth_login(), imap_login(), pop_pass(), and portal_endpoint_auth().

4.5.2.8 stringer_t* auth_t::nonce

Definition at line 57 of file auth.h.

Referenced by auth_challenge(), auth_free(), and auth_response().

4.5.2.9 stringer_t* auth_t::salt

Definition at line 56 of file auth.h.

Referenced by api_endpoint_auth(), auth_data_fetch(), auth_free(), auth_login(), auth_response(), imap_login(), pop_pass(), and portal_endpoint_auth().

4.5.2.10 struct { ... } auth_t::seasoning

Referenced by api_endpoint_auth(), auth_challenge(), auth_data_fetch(), auth_free(), auth_login(), auth_response(), imap_login(), pop_pass(), and portal_endpoint_auth().

4.5.2.11 struct { ... } auth_t::status

Referenced by auth_data_fetch(), auth_login(), imap_login(), pop_pass(), portal_endpoint_auth(), smtp_auth_login(), and smtp_auth_plain().

4.5.2.12 stringer_t* auth_t::token

Definition at line 71 of file auth.h.

Referenced by auth_data_fetch(), auth_free(), and auth_login().

4.5.2.13 struct { ... } auth_t::tokens

Referenced by api_endpoint_auth(), auth_challenge(), auth_data_fetch(), auth_free(), auth_login(), auth_response(), imap_login(), pop_pass(), portal_endpoint_auth(), smtp_auth_login(), smtp_auth_plain(), and teacher_process().

4.5.2.14 stringer_t* auth_t::username

Definition at line 48 of file auth.h.

Referenced by api_endpoint_auth(), auth_challenge(), auth_data_fetch(), auth_free(), auth_login(), auth_response(), imap_login(), pop_pass(), portal_endpoint_auth(), smtp_auth_login(), and smtp_auth_plain().

4.5.2.15 uint64_t auth_t::usernum

Definition at line 47 of file auth.h.

Referenced by api_endpoint_auth(), auth_data_fetch(), auth_login(), imap_login(), pop_pass(), and portal_endpoint_auth().

4.5.2.16 stringer_t* auth_t::verification

Definition at line 66 of file auth.h.

Referenced by api_endpoint_auth(), auth_challenge(), auth_data_fetch(), auth_free(), auth_login(), auth_response(), imap_login(), pop_pass(), portal_endpoint_auth(), smtp_auth_login(), smtp_auth_plain(), and teacher_process().

4.6 batch_heap_t Struct Reference

```
#include <ed25519-donna-batchverify.h>
```

Data Fields

- unsigned char `r` [heap_batch_size][16]
- `ge25519` points [heap_batch_size]
- `bignum256modm` scalars [heap_batch_size]
- `heap_index_t` heap [heap_batch_size]
- `size_t` size

4.6.1 Detailed Description

Definition at line 13 of file ed25519-donna-batchverify.h.

4.6.2 Field Documentation

4.6.2.1 heap_index_t batch_heap_t::heap[heap_batch_size]

Definition at line 17 of file ed25519-donna-batchverify.h.

4.6.2.2 ge25519 batch_heap_t::points[heap_batch_size]

Definition at line 15 of file ed25519-donna-batchverify.h.

Referenced by ed25519_sign_open_batch().

4.6.2.3 unsigned char batch_heap_t::r[heap_batch_size][16]

Definition at line 14 of file ed25519-donna-batchverify.h.

Referenced by ed25519_sign_open_batch().

4.6.2.4 bignum256modm batch_heap_t::scalars[heap_batch_size]

Definition at line 16 of file ed25519-donna-batchverify.h.

Referenced by ed25519_sign_open_batch().

4.6.2.5 size_t batch_heap_t::size

Definition at line 18 of file ed25519-donna-batchverify.h.

4.7 cache_keys_t Struct Reference

```
#include <cache.h>
```

Data Fields

- [size_t offset](#)
- [multi_t norm](#)
- [char * name](#)
- [char * description](#)
- [bool_t required](#)

4.7.1 Detailed Description

Definition at line 11 of file `cache.h`.

4.7.2 Field Documentation

4.7.2.1 `char* cache_keys_t::description`

Definition at line 15 of file `cache.h`.

4.7.2.2 `char* cache_keys_t::name`

Definition at line 14 of file `cache.h`.

Referenced by `cache_set_value()`.

4.7.2.3 `multi_t cache_keys_t::norm`

Definition at line 13 of file `cache.h`.

Referenced by `cache_free()`, `cache_output_help()`, `cache_output_settings()`, `cache_set_value()`, and `cache_validate()`.

4.7.2.4 `size_t cache_keys_t::offset`

Definition at line 12 of file `cache.h`.

Referenced by `cache_free()`, `cache_output_settings()`, `cache_set_value()`, and `cache_validate()`.

4.7.2.5 `bool_t cache_keys_t::required`

Definition at line 16 of file `cache.h`.

Referenced by `cache_output_help()`.

4.8 cache_t Struct Reference

```
#include <cache.h>
```

Data Fields

- char * [name](#)
- uint32_t [port](#)
- uint32_t [weight](#)

4.8.1 Detailed Description

Definition at line 19 of file cache.h.

4.8.2 Field Documentation

4.8.2.1 char* cache_t::name

Definition at line 20 of file cache.h.

4.8.2.2 uint32_t cache_t::port

Definition at line 21 of file cache.h.

4.8.2.3 uint32_t cache_t::weight

Definition at line 22 of file cache.h.

4.9 cached_object Struct Reference

```
#include <cache.h>
```

Data Fields

- `time_t timestamp`
The UTC timestamp for when this object was cached. Used with ttl.
- `unsigned char id [32]`
The identifier of the cached item as a SHA-256 hash.
- `cached_data_type_t dtype`
The type of the cached data being stored.
- `unsigned long ttl`
Optional time-to-live in seconds.
- `time_t expiration`
The time when the record will expire, in UTC.
- `int relaxed`
Whether or not the cache follows a relaxed eviction policy.
- `void * data`
The data associated with the cached object.
- `struct cached_object * prev`
A pointer to the previous cached object in the linked list.
- `struct cached_object * next`
A pointer to the next cached object in the linked list.
- `unsigned char persists`
Not everything in the cache should be persisted.
- `struct cached_object * shadow`

4.9.1 Detailed Description

Definition at line 34 of file `cache.h`.

4.9.2 Field Documentation

4.9.2.1 `void* cached_object::data`

The data associated with the cached object.

Definition at line 41 of file `cache.h`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_cached_object_exists_cmp()`, `_clone_cached_object()`, `_create_cached_object()`, `_destroy_cache_entry()`, `_do_ocsp_validation()`, `_dump_cache()`, `_dump_cache_data()`, `_find_cached_object_cmp()`, `_fixup_dnskey_validation()`, `_get_cache_obj_data()`, `_get_dime_record()`, `_load_cache_contents()`, `_remove_cached_object_cmp()`, `_save_cache_contents()`, and `_unlink_object()`.

4.9.2.2 `cached_data_type_t` `cached_object::dtype`

The type of the cached data being stored.

Definition at line 37 of file `cache.h`.

Referenced by `_add_cached_object()`, `_clone_cached_object()`, `_create_cached_object()`, `_destroy_cache_entry()`, `_dump_cache_data()`, `_get_cache_obj_data()`, `_load_cache_contents()`, `_replace_object()`, and `_unlink_object()`.

4.9.2.3 `time_t` `cached_object::expiration`

The time when the record will expire, in UTC.

Definition at line 39 of file `cache.h`.

Referenced by `_add_cached_object()`, `_clone_cached_object()`, `_create_cached_object()`, `_dump_cache()`, `_get_dime_record()`, `_is_object_expired()`, and `_load_cache_contents()`.

4.9.2.4 `unsigned char` `cached_object::id[32]`

The identifier of the cached item as a SHA-256 hash.

Definition at line 36 of file `cache.h`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_cached_object_exists()`, `_clone_cached_object()`, `_find_cached_object()`, `_load_cache_contents()`, `_remove_cached_object()`, and `_replace_object()`.

4.9.2.5 `struct cached_object*` `cached_object::next` `[read]`

A pointer to the next cached object in the linked list.

Definition at line 43 of file `cache.h`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_dump_cache()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_fixup_dnskey_validation()`, `_load_cache_contents()`, `_remove_cached_object()`, `_remove_cached_object_cmp()`, `_replace_object()`, `_save_cache_contents()`, and `_unlink_object()`.

4.9.2.6 `unsigned char` `cached_object::persists`

Not everything in the cache should be persisted.

Definition at line 44 of file `cache.h`.

Referenced by `_add_cached_object()`, `_clone_cached_object()`, `_create_cached_object()`, `_dump_cache()`, `_fixup_dnskey_validation()`, `_load_cache_contents()`, and `_save_cache_contents()`.

4.9.2.7 `struct cached_object*` `cached_object::prev` `[read]`

A pointer to the previous cached object in the linked list.

Definition at line 42 of file `cache.h`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_replace_object()`, and `_unlink_object()`.

4.9.2.8 `int` `cached_object::relaxed`

Whether or not the cache follows a relaxed eviction policy.

Definition at line 40 of file `cache.h`.

Referenced by `_add_cached_object()`, `_clone_cached_object()`, `_create_cached_object()`, `_dump_cache()`, and `_is_object_expired()`.

4.9.2.9 `struct cached_object* cached_object::shadow` [read]

A saved copy of the "real" cache entry to be saved, if this cached object is merely temporarily overriding it.

Definition at line 45 of file `cache.h`.

Referenced by `_dump_cache()`, `_replace_object()`, and `_save_cache_contents()`.

4.9.2.10 `time_t cached_object::timestamp`

The UTC timestamp for when this object was cached. Used with `ttl`.

Definition at line 35 of file `cache.h`.

Referenced by `_add_cached_object()`, `_clone_cached_object()`, `_create_cached_object()`, `_dump_cache()`, `_is_object_expired()`, and `_load_cache_contents()`.

4.9.2.11 `unsigned long cached_object::ttl`

Optional time-to-live in seconds.

Definition at line 38 of file `cache.h`.

Referenced by `_add_cached_object()`, `_clone_cached_object()`, `_create_cached_object()`, `_dump_cache()`, `_is_object_expired()`, and `_load_cache_contents()`.

4.10 cached_store_t Struct Reference

```
#include <cache.h>
```

Data Fields

- [cached_data_type_t dtype](#)
The type of data that will be stored within.
- `const char *` [description](#)
A text description of the cache store.
- `unsigned char` [internal](#)
- `cached_object_t *` [head](#)
A pointer to the head of the cached object list.
- `pthread_mutex_t` [lock](#)
For multi-threaded safe code.
- `void(*` [destructor](#) `)(void *)`
A function pointer to a destructor used to free cached object data.
- `void *(*` [serialize](#) `)(void *, size_t *)`
A function pointer to a routine used to serialize cached data.
- `void *(*` [deserialize](#) `)(void *, size_t)`
A function pointer to a routine used to deserialize cached data.
- `void(*` [dump](#) `)(FILE *, void *, int)`
A function pointer to a routine used to dump the cached data.
- `void *(*` [clone](#) `)(void *)`

4.10.1 Detailed Description

Definition at line 50 of file cache.h.

4.10.2 Field Documentation

4.10.2.1 `void(* cached_store_t::clone)(void *)`

An optional pointer to a routine that can be used to clone data. If not specified, the serialize and deserialize routine will be used together to recreate the functionality of this function.

Referenced by `_clone_cached_object()`.

4.10.2.2 `const char* cached_store_t::description`

A text description of the cache store.

Definition at line 52 of file cache.h.

4.10.2.3 void>(* cached_store_t::deserialize)(void *, size_t)

A function pointer to a routine used to deserialize cached data.

Referenced by _clone_cached_object(), and _load_cache_contents().

4.10.2.4 void(* cached_store_t::destructor)(void *)

A function pointer to a destructor used to free cached object data.

Referenced by _destroy_cache_entry(), and _unlink_object().

4.10.2.5 cached_data_type_t cached_store_t::dtype

The type of data that will be stored within.

Definition at line 51 of file cache.h.

Referenced by _add_cached_object(), _add_cached_object_cmp(), _add_cached_object_cmp_forced(), _add_cached_object_forced(), and _dump_cache().

4.10.2.6 void(* cached_store_t::dump)(FILE *, void *, int)

A function pointer to a routine used to dump the cached data.

Referenced by _dump_cache_data(), and _unlink_object().

4.10.2.7 cached_object_t* cached_store_t::head

A pointer to the head of the cached object list.

Definition at line 55 of file cache.h.

Referenced by _add_cached_object(), _add_cached_object_cmp(), _cached_object_exists(), _cached_object_exists_cmp(), _find_cached_object(), _find_cached_object_cmp(), _fixup_dnskey_validation(), _load_cache_contents(), _remove_cached_object(), _remove_cached_object_cmp(), _replace_object(), _save_cache_contents(), and _unlink_object().

4.10.2.8 unsigned char cached_store_t::internal

Determines whether the cached store is for internal use only. If it is, objects will not be cloned before being returned to the caller.

Definition at line 53 of file cache.h.

Referenced by _clone_cached_object(), and _get_cache_obj_data().

4.10.2.9 pthread_mutex_t cached_store_t::lock

For multi-threaded safe code.

Definition at line 56 of file cache.h.

Referenced by _lock_cache_store(), and _unlock_cache_store().

4.10.2.10 void(* cached_store_t::serialize)(void *, size_t *)

A function pointer to a routine used to serialize cached data.

Referenced by _clone_cached_object().

4.11 derror_t Struct Reference

```
#include <error_codes.h>
```

Data Fields

- [dime_errcode_t](#) code
- char const * [message](#)

4.11.1 Detailed Description

Definition at line 12 of file error_codes.h.

4.11.2 Field Documentation

4.11.2.1 `dime_errcode_t` `derror_t::code`

Definition at line 13 of file error_codes.h.

4.11.2.2 `char const*` `derror_t::message`

Definition at line 14 of file error_codes.h.

4.12 dime_ctx Struct Reference

Data Fields

- [log_function_t log_callback](#)

4.12.1 Detailed Description

Definition at line 57 of file dime_ctx.c.

4.12.2 Field Documentation

4.12.2.1 log_function_t dime_ctx::log_callback

Definition at line 58 of file dime_ctx.c.

Referenced by dime_ctx_log().

4.13 dime_record_t Struct Reference

```
#include <mrec.h>
```

Data Fields

- unsigned short [version](#)
The DIME management record syntax version.
- unsigned char ** [pubkey](#)
An array of 32-byte ED25519 organizational public key(s) (POKs).
- unsigned char ** [tlssig](#)
An array of 64-byte TLS signature(s) for MX (or DX/KS) by the POK(s).
- [dime_msg_policy](#) policy
Policy for sending/accepting messages.
- char * [syndicates](#)
Alternative authoritative signet lookup sources.
- char ** [dx](#)
An array of CNAME(s) for DIME delivery host (if not present, then MX).
- unsigned long [expiry](#)
Number of days before a cached management record is discarded.
- [dime_sub_policy](#) subdomain
Determines whether subdomains will have authority over their own records.
- int [validated](#)
1 if validated by DNSSEC; 0 if not DNSSEC-protected; -1 if DNSSEC sig failed.

4.13.1 Detailed Description

Definition at line 32 of file mrec.h.

4.13.2 Field Documentation

4.13.2.1 char** dime_record_t::dx

An array of CNAME(s) for DIME delivery host (if not present, then MX).

Definition at line 38 of file mrec.h.

Referenced by `_deserialize_dime_record_cb()`, `_destroy_dime_record()`, `_dump_dime_record_cb()`, `_parse_dime_record()`, `_serialize_dime_record_cb()`, and `_sgnt_resolv_dmtpl_connect()`.

4.13.2.2 unsigned long dime_record_t::expiry

Number of days before a cached management record is discarded.

Definition at line 39 of file mrec.h.

Referenced by _deserialize_dime_record_cb(), _dump_dime_record_cb(), _get_dime_record(), and _serialize_dime_record_cb().

4.13.2.3 dime_msg_policy dime_record_t::policy

Policy for sending/accepting messages.

Definition at line 36 of file mrec.h.

Referenced by _deserialize_dime_record_cb(), _dump_dime_record_cb(), _parse_dime_record(), _serialize_dime_record_cb(), and _validate_dime_record().

4.13.2.4 unsigned char** dime_record_t::pubkey

An array of 32-byte ED25519 organizational public key(s) (POKs).

Definition at line 34 of file mrec.h.

Referenced by _deserialize_dime_record_cb(), _destroy_dime_record(), _dump_dime_record_cb(), _get_signet(), _parse_dime_record(), _serialize_dime_record_cb(), _validate_dime_record(), and _verify_dx_certificate().

4.13.2.5 dime_sub_policy dime_record_t::subdomain

Determines whether subdomains will have authority over their own records.

Definition at line 40 of file mrec.h.

Referenced by _deserialize_dime_record_cb(), _dump_dime_record_cb(), _parse_dime_record(), _serialize_dime_record_cb(), and _validate_dime_record().

4.13.2.6 char* dime_record_t::syndicates

Alternative authoritative signet lookup sources.

Definition at line 37 of file mrec.h.

Referenced by _deserialize_dime_record_cb(), _destroy_dime_record(), _dump_dime_record_cb(), and _serialize_dime_record_cb().

4.13.2.7 unsigned char** dime_record_t::tlssig

An array of 64-byte TLS signature(s) for MX (or DX/KS) by the POK(s).

Definition at line 35 of file mrec.h.

Referenced by _deserialize_dime_record_cb(), _destroy_dime_record(), _dump_dime_record_cb(), _parse_dime_record(), _serialize_dime_record_cb(), and _verify_dx_certificate().

4.13.2.8 int dime_record_t::validated

1 if validated by DNSSEC; 0 if not DNSSEC-protected; -1 if DNSSEC sig failed.

Definition at line 41 of file mrec.h.

Referenced by _deserialize_dime_record_cb(), _dump_dime_record_cb(), _get_dime_record(), _serialize_dime_record_cb(), _sgnt_resolve_dmtpt_connect(), and _verify_dx_certificate().

4.13.2.9 unsigned short dime_record_t::version

The DIME management record syntax version.

Definition at line 33 of file mrec.h.

Referenced by `_deserialize_dime_record_cb()`, `_dump_dime_record_cb()`, `_parse_dime_record()`, `_serialize_dime_record_cb()`, and `_validate_dime_record()`.

4.14 dmime_chunk_key_t Struct Reference

```
#include <common.h>
```

Data Fields

- unsigned int [required](#)
- unsigned int [unique](#)
- unsigned int [encrypted](#)
- unsigned int [sequential](#)
- [dmime_chunk_section_t](#) section
- [dmime_payload_type_t](#) payload
- unsigned int [auth_keyslot](#)
- unsigned int [orig_keyslot](#)
- unsigned int [dest_keyslot](#)
- unsigned int [recp_keyslot](#)
- const char * [name](#)
- const char * [description](#)

4.14.1 Detailed Description

Definition at line 120 of file common.h.

4.14.2 Field Documentation

4.14.2.1 unsigned int dmime_chunk_key_t::auth_keyslot

Definition at line 129 of file common.h.

4.14.2.2 const char* dmime_chunk_key_t::description

Definition at line 135 of file common.h.

4.14.2.3 unsigned int dmime_chunk_key_t::dest_keyslot

Definition at line 131 of file common.h.

4.14.2.4 unsigned int dmime_chunk_key_t::encrypted

Definition at line 123 of file common.h.

4.14.2.5 const char* dmime_chunk_key_t::name

Definition at line 134 of file common.h.

4.14.2.6 unsigned int dmime_chunk_key_t::orig_keyslot

Definition at line 130 of file common.h.

4.14.2.7 dmime_payload_type_t dmime_chunk_key_t::payload

Definition at line 127 of file common.h.

4.14.2.8 unsigned int dmime_chunk_key_t::recp_keyslot

Definition at line 132 of file common.h.

4.14.2.9 unsigned int dmime_chunk_key_t::required

Definition at line 121 of file common.h.

4.14.2.10 dmime_chunk_section_t dmime_chunk_key_t::section

Definition at line 126 of file common.h.

4.14.2.11 unsigned int dmime_chunk_key_t::sequential

Definition at line 124 of file common.h.

4.14.2.12 unsigned int dmime_chunk_key_t::unique

Definition at line 122 of file common.h.

4.15 dmime_common_headers_t Struct Reference

```
#include <common.h>
```

Data Fields

- [sds headers](#) [DMIME_NUM_COMMON_HEADERS]

4.15.1 Detailed Description

Definition at line 24 of file common.h.

4.15.2 Field Documentation

4.15.2.1 sds dmime_common_headers_t::headers[DMIME_NUM_COMMON_HEADERS]

Definition at line 25 of file common.h.

4.16 dmime_envelope_object_t Struct Reference

```
#include <parse.h>
```

Data Fields

- [sds auth_recip](#)
- [sds auth_recip_fp](#)
- [sds dest_orig](#)
- [sds dest_orig_fp](#)

4.16.1 Detailed Description

Definition at line 12 of file parse.h.

4.16.2 Field Documentation

4.16.2.1 sds dmime_envelope_object_t::auth_recip

Definition at line 13 of file parse.h.

4.16.2.2 sds dmime_envelope_object_t::auth_recip_fp

Definition at line 14 of file parse.h.

4.16.2.3 sds dmime_envelope_object_t::dest_orig

Definition at line 15 of file parse.h.

4.16.2.4 sds dmime_envelope_object_t::dest_orig_fp

Definition at line 16 of file parse.h.

4.17 dmime_header_key_t Struct Reference

```
#include <parse.h>
```

Data Fields

- int [required](#)
- const char * [label](#)
- size_t [label_length](#)

4.17.1 Detailed Description

Definition at line 6 of file parse.h.

4.17.2 Field Documentation

4.17.2.1 const char* dmime_header_key_t::label

Definition at line 8 of file parse.h.

4.17.2.2 size_t dmime_header_key_t::label_length

Definition at line 9 of file parse.h.

4.17.2.3 int dmime_header_key_t::required

Definition at line 7 of file parse.h.

4.18 dmime_message_t Struct Reference

```
#include <crypto.h>
```

Data Fields

- [dime_number_t](#) dime_num
- [uint32_t](#) size
- [dmime_tracing_t](#) * tracing
- [dmime_message_chunk_t](#) * ephemeral
- [dmime_message_chunk_t](#) * origin
- [dmime_message_chunk_t](#) * destination
- [dmime_message_chunk_t](#) * common_headers
- [dmime_message_chunk_t](#) * other_headers
- [dmime_message_chunk_t](#) ** display
- [dmime_message_chunk_t](#) ** attach
- [dmime_message_chunk_t](#) * author_tree_sig
- [dmime_message_chunk_t](#) * author_full_sig
- [dmime_message_chunk_t](#) * origin_meta_bounce_sig
- [dmime_message_chunk_t](#) * origin_display_bounce_sig
- [dmime_message_chunk_t](#) * origin_full_sig
- [dmime_message_state_t](#) state

4.18.1 Detailed Description

Definition at line 117 of file `crypto.h`.

4.18.2 Field Documentation

4.18.2.1 `dmime_message_chunk_t** dmime_message_t::attach`

Definition at line 137 of file `crypto.h`.

4.18.2.2 `dmime_message_chunk_t* dmime_message_t::author_full_sig`

Definition at line 141 of file `crypto.h`.

4.18.2.3 `dmime_message_chunk_t* dmime_message_t::author_tree_sig`

Definition at line 139 of file `crypto.h`.

4.18.2.4 `dmime_message_chunk_t* dmime_message_t::common_headers`

Definition at line 131 of file `crypto.h`.

4.18.2.5 `dmime_message_chunk_t* dmime_message_t::destination`

Definition at line 129 of file `crypto.h`.

4.18.2.6 dmime_number_t dmime_message_t::dime_num

Definition at line 119 of file crypto.h.

4.18.2.7 dmime_message_chunk_t dmime_message_t::display**

Definition at line 135 of file crypto.h.

4.18.2.8 dmime_message_chunk_t* dmime_message_t::ephemeral

Definition at line 125 of file crypto.h.

4.18.2.9 dmime_message_chunk_t* dmime_message_t::origin

Definition at line 127 of file crypto.h.

4.18.2.10 dmime_message_chunk_t* dmime_message_t::origin_display_bounce_sig

Definition at line 145 of file crypto.h.

4.18.2.11 dmime_message_chunk_t* dmime_message_t::origin_full_sig

Definition at line 147 of file crypto.h.

4.18.2.12 dmime_message_chunk_t* dmime_message_t::origin_meta_bounce_sig

Definition at line 143 of file crypto.h.

4.18.2.13 dmime_message_chunk_t* dmime_message_t::other_headers

Definition at line 133 of file crypto.h.

4.18.2.14 uint32_t dmime_message_t::size

Definition at line 121 of file crypto.h.

4.18.2.15 dmime_message_state_t dmime_message_t::state

Definition at line 149 of file crypto.h.

4.18.2.16 dmime_tracing_t* dmime_message_t::tracing

Definition at line 123 of file crypto.h.

4.19 dmime_object_t Struct Reference

```
#include <crypto.h>
```

Data Fields

- [dmime_actor_t actor](#)
- [sds author](#)
- [sds recipient](#)
- [sds origin](#)
- [sds destination](#)
- [sds fp_author](#)
- [sds fp_recipient](#)
- [sds fp_origin](#)
- [sds fp_destination](#)
- [signet_t * signet_author](#)
- [signet_t * signet_recipient](#)
- [signet_t * signet_origin](#)
- [signet_t * signet_destination](#)
- [dmime_common_headers_t * common_headers](#)
- [sds other_headers](#)
- [dmime_object_chunk_t * display](#)
- [dmime_object_chunk_t * attach](#)
- [dmime_object_state_t state](#)

4.19.1 Detailed Description

Definition at line 59 of file `crypto.h`.

4.19.2 Field Documentation

4.19.2.1 dmime_actor_t dmime_object_t::actor

Definition at line 61 of file `crypto.h`.

4.19.2.2 dmime_object_chunk_t* dmime_object_t::attach

Definition at line 84 of file `crypto.h`.

4.19.2.3 sds dmime_object_t::author

Definition at line 63 of file `crypto.h`.

4.19.2.4 dmime_common_headers_t* dmime_object_t::common_headers

Definition at line 79 of file `crypto.h`.

4.19.2.5 sds dmime_object_t::destination

Definition at line 67 of file `crypto.h`.

4.19.2.6 dmime_object_chunk_t* dmime_object_t::display

Definition at line 83 of file crypto.h.

4.19.2.7 sds dmime_object_t::fp_author

Definition at line 69 of file crypto.h.

4.19.2.8 sds dmime_object_t::fp_destination

Definition at line 72 of file crypto.h.

4.19.2.9 sds dmime_object_t::fp_origin

Definition at line 71 of file crypto.h.

4.19.2.10 sds dmime_object_t::fp_recipient

Definition at line 70 of file crypto.h.

4.19.2.11 sds dmime_object_t::origin

Definition at line 66 of file crypto.h.

4.19.2.12 sds dmime_object_t::other_headers

Definition at line 81 of file crypto.h.

4.19.2.13 sds dmime_object_t::recipient

Definition at line 64 of file crypto.h.

4.19.2.14 signet_t* dmime_object_t::signet_author

Definition at line 74 of file crypto.h.

4.19.2.15 signet_t* dmime_object_t::signet_destination

Definition at line 77 of file crypto.h.

4.19.2.16 signet_t* dmime_object_t::signet_origin

Definition at line 76 of file crypto.h.

4.19.2.17 signet_t* dmime_object_t::signet_recipient

Definition at line 75 of file crypto.h.

4.19.2.18 dmime_object_state_t dmime_object_t::state

Definition at line 86 of file crypto.h.

4.20 dmtp_argument_t Struct Reference

```
#include <commands.h>
```

Data Fields

- char const * [arg_name](#)
- size_t [arg_name_len](#)
- [dmtp_argument_type_t](#) type
- size_t [size](#)

4.20.1 Detailed Description

Definition at line 50 of file commands.h.

4.20.2 Field Documentation

4.20.2.1 char const* dmtp_argument_t::arg_name

Definition at line 51 of file commands.h.

4.20.2.2 size_t dmtp_argument_t::arg_name_len

Definition at line 52 of file commands.h.

4.20.2.3 size_t dmtp_argument_t::size

Definition at line 54 of file commands.h.

4.20.2.4 dmtp_argument_type_t dmtp_argument_t::type

Definition at line 53 of file commands.h.

4.21 dmtp_command_key_t Struct Reference

```
#include <commands.h>
```

Data Fields

- char const * [com_name](#)
- size_t [com_name_len](#)
- [dmtp_argument_t](#) [args](#) [DMTP_MAX_ARGUMENT_NUM]

4.21.1 Detailed Description

Definition at line 62 of file commands.h.

4.21.2 Field Documentation

4.21.2.1 dmtp_argument_t dmtp_command_key_t::args[DMTP_MAX_ARGUMENT_NUM]

Definition at line 65 of file commands.h.

4.21.2.2 char const* dmtp_command_key_t::com_name

Definition at line 63 of file commands.h.

4.21.2.3 size_t dmtp_command_key_t::com_name_len

Definition at line 64 of file commands.h.

4.22 dmtplib_command_t Struct Reference

```
#include <commands.h>
```

Data Fields

- [dmtplib_command_type_t](#) type
- [sds](#) args [DMTP_MAX_ARGUMENT_NUM]

4.22.1 Detailed Description

Definition at line 57 of file commands.h.

4.22.2 Field Documentation

4.22.2.1 [sds](#) dmtplib_command_t::args[DMTP_MAX_ARGUMENT_NUM]

Definition at line 59 of file commands.h.

Referenced by [dime_dmtplib_command_ahlo\(\)](#), [dime_dmtplib_command_helo\(\)](#), [dime_dmtplib_command_hist\(\)](#), [dime_dmtplib_command_mail\(\)](#), [dime_dmtplib_command_noop\(\)](#), [dime_dmtplib_command_rcpt\(\)](#), [dime_dmtplib_command_sant_domain\(\)](#), [dime_dmtplib_command_sant_user\(\)](#), [dime_dmtplib_command_starttls\(\)](#), [dime_dmtplib_command_vrfy_domain\(\)](#), and [dime_dmtplib_command_vrfy_user\(\)](#).

4.22.2.2 [dmtplib_command_type_t](#) dmtplib_command_t::type

Definition at line 58 of file commands.h.

4.23 dmtplib_session_t Struct Reference

```
#include <dmtplib.h>
```

Data Fields

- char * [domain](#)
The name of the dark domain underlying the DMTP connection.
- char * [dx](#)
The canonical name of the DX that we're connected to.
- SSL * [con](#)
The handle to this DMTP session's underlying SSL connection.
- [dime_record_t](#) * [drec](#)
The DIME management record associated with this dark domain.
- [dmtplib_mode_t](#) [mode](#)
The current mode of this connection (if made through dual mode).
- unsigned int [active](#)
Boolean flag: whether or not this session is active.
- int [_fd](#)
- unsigned char [_inbuf](#) [DMTP_LINE_BUF_SIZE+1]
- size_t [_inpos](#)

4.23.1 Detailed Description

Definition at line 12 of file dmtplib.h.

4.23.2 Field Documentation

4.23.2.1 int dmtplib_session_t::_fd

Definition at line 37 of file dmtplib.h.

Referenced by [_dx_connect_dual\(\)](#), [_dx_connect_standard\(\)](#), [_sgnt_resolv_destroy_dmtplib_session\(\)](#), [_sgnt_resolv_dmtplib_initiate_starttls\(\)](#), [_sgnt_resolv_dmtplib_issue_command\(\)](#), [_sgnt_resolv_dmtplib_quit\(\)](#), [_sgnt_resolv_dmtplib_write_data\(\)](#), and [_sgnt_resolv_read_dmtplib_line\(\)](#).

4.23.2.2 unsigned char dmtplib_session_t::_inbuf[DMTP_LINE_BUF_SIZE+1]

Definition at line 38 of file dmtplib.h.

Referenced by [_sgnt_resolv_destroy_dmtplib_session\(\)](#), and [_sgnt_resolv_read_dmtplib_line\(\)](#).

4.23.2.3 size_t dmtplib_session_t::_inpos

Definition at line 39 of file dmtplib.h.

Referenced by [_sgnt_resolv_read_dmtplib_line\(\)](#).

4.23.2.4 unsigned int dmtplib_session_t::active

Boolean flag: whether or not this session is active.

Definition at line 35 of file dmtplib.h.

Referenced by _dx_connect_dual(), _sgnt_resolv_dmtplib_quit(), and _sgnt_resolv_read_dmtplib_line().

4.23.2.5 SSL* dmtplib_session_t::con

The handle to this DMTP session's underlying SSL connection.

Definition at line 32 of file dmtplib.h.

Referenced by _dx_connect_standard(), _sgnt_resolv_destroy_dmtplib_session(), _sgnt_resolv_dmtplib_initiate_starttls(), _sgnt_resolv_dmtplib_issue_command(), _sgnt_resolv_dmtplib_quit(), _sgnt_resolv_dmtplib_write_data(), _sgnt_resolv_read_dmtplib_line(), and _verify_dx_certificate().

4.23.2.6 char* dmtplib_session_t::domain

The name of the dark domain underlying the DMTP connection.

Definition at line 30 of file dmtplib.h.

Referenced by _dx_connect_dual(), _dx_connect_standard(), and _sgnt_resolv_destroy_dmtplib_session().

4.23.2.7 dime_record_t* dmtplib_session_t::drec

The DIME management record associated with this dark domain.

Definition at line 33 of file dmtplib.h.

Referenced by _dx_connect_dual(), _dx_connect_standard(), _get_sigmet(), _sgnt_resolv_destroy_dmtplib_session(), and _verify_dx_certificate().

4.23.2.8 char* dmtplib_session_t::dx

The canonical name of the DX that we're connected to.

Definition at line 31 of file dmtplib.h.

Referenced by _dx_connect_dual(), _dx_connect_standard(), _sgnt_resolv_destroy_dmtplib_session(), and _verify_dx_certificate().

4.23.2.9 dmtplib_mode_t dmtplib_session_t::mode

The current mode of this connection (if made through dual mode).

Definition at line 34 of file dmtplib.h.

Referenced by _dx_connect_dual(), _dx_connect_standard(), _sgnt_resolv_dmtplib_initiate_starttls(), and _sgnt_resolv_read_dmtplib_line().

4.24 dnskey Struct Reference

```
#include <dns.h>
```

Data Fields

- char * [label](#)
- unsigned char [algorithm](#)
- unsigned char [is_zone](#)
- unsigned char [is_sep](#)
- RSA * [pubkey](#)
- unsigned int [keytag](#)
- size_t [rdlen](#)
- unsigned char * [rdata](#)
- [dnskey_t](#) ** [signkeys](#)

The DNSKEY(s) that were used to sign the RRSIG guaranteeing this DNSKEY.

- [ds_t](#) ** [dse](#)

The DS RR(s) that match this DNSKEY record.

- unsigned int [validated](#)

Has this key been validated?

- unsigned int [do_cache](#)

If set, this DNSKEY will be persisted IFF it has been validated.

4.24.1 Detailed Description

Definition at line 105 of file dns.h.

4.24.2 Field Documentation

4.24.2.1 unsigned char dnskey::algorithm

Definition at line 107 of file dns.h.

Referenced by [_add_dnskey_entry_rsa\(\)](#), [_clone_dnskey_record_cb\(\)](#), [_deserialize_dnskey_record_cb\(\)](#), [_dump_dnskey_record_cb\(\)](#), [_get_ds_by_dnskey\(\)](#), and [_serialize_dnskey_record_cb\(\)](#).

4.24.2.2 unsigned int dnskey::do_cache

If set, this DNSKEY will be persisted IFF it has been validated.

Definition at line 117 of file dns.h.

Referenced by [_add_dnskey_entry_rsa\(\)](#), and [_fixup_dnskey_validation\(\)](#).

4.24.2.3 ds_t** dnskey::dse

The DS RR(s) that match this DNSKEY record.

Definition at line 115 of file dns.h.

Referenced by [_clone_dnskey_record_cb\(\)](#), [_destroy_dnskey\(\)](#), [_dump_dnskey_record_cb\(\)](#), and [_lookup_ds\(\)](#).

4.24.2.4 unsigned char dnskey::is_sep

Definition at line 109 of file dns.h.

Referenced by `_add_dnskey_entry_rsa()`, `_clone_dnskey_record_cb()`, `_deserialize_dnskey_record_cb()`, `_dump_dnskey_record_cb()`, and `_serialize_dnskey_record_cb()`.

4.24.2.5 unsigned char dnskey::is_zone

Definition at line 108 of file dns.h.

Referenced by `_add_dnskey_entry_rsa()`, `_clone_dnskey_record_cb()`, `_deserialize_dnskey_record_cb()`, `_dump_dnskey_record_cb()`, and `_serialize_dnskey_record_cb()`.

4.24.2.6 unsigned int dnskey::keytag

Definition at line 111 of file dns.h.

Referenced by `_add_dnskey_entry_rsa()`, `_clone_dnskey_record_cb()`, `_deserialize_dnskey_record_cb()`, `_dnskey_tag_comparator()`, `_dump_dnskey_record_cb()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, and `_serialize_dnskey_record_cb()`.

4.24.2.7 char* dnskey::label

Definition at line 106 of file dns.h.

Referenced by `_add_dnskey_entry_rsa()`, `_clone_dnskey_record_cb()`, `_compute_dnskey_sha_hash()`, `_deserialize_dnskey_record_cb()`, `_destroy_dnskey()`, `_dnskey_domain_comparator()`, `_dnskey_tag_comparator()`, `_dump_dnskey_record_cb()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, `_is_validated_key()`, `_load_dnskey_file()`, and `_serialize_dnskey_record_cb()`.

4.24.2.8 RSA* dnskey::pubkey

Definition at line 110 of file dns.h.

Referenced by `_add_dnskey_entry_rsa()`, `_clone_dnskey_record_cb()`, `_deserialize_dnskey_record_cb()`, `_destroy_dnskey()`, `_serialize_dnskey_record_cb()`, and `_validate_rrsig_rr()`.

4.24.2.9 unsigned char* dnskey::rdata

Definition at line 113 of file dns.h.

Referenced by `_add_dnskey_entry_rsa()`, `_clone_dnskey_record_cb()`, `_compute_dnskey_sha_hash()`, `_deserialize_dnskey_record_cb()`, `_destroy_dnskey()`, and `_serialize_dnskey_record_cb()`.

4.24.2.10 size_t dnskey::rdlen

Definition at line 112 of file dns.h.

Referenced by `_add_dnskey_entry_rsa()`, `_clone_dnskey_record_cb()`, `_compute_dnskey_sha_hash()`, `_deserialize_dnskey_record_cb()`, `_dump_dnskey_record_cb()`, and `_serialize_dnskey_record_cb()`.

4.24.2.11 dnskey_t** dnskey::signkeys

The DNSKEY(s) that were used to sign the RRSIG guaranteeing this DNSKEY.

Definition at line 114 of file dns.h.

Referenced by `_clone_dnskey_record_cb()`, `_destroy_dnskey()`, `_dump_dnskey_record_cb()`, and `_is_validated_key()`.

4.24.2.12 unsigned int dnskey::validated

Has this key been validated?

Definition at line 116 of file dns.h.

Referenced by `_clone_dnskey_record_cb()`, `_deserialize_dnskey_record_cb()`, `_fixup_dnskey_validation()`, `_is_validated_key()`, `_load_dnskey_file()`, and `_serialize_dnskey_record_cb()`.

4.25 ds Struct Reference

```
#include <dns.h>
```

Data Fields

- char * [label](#)
- unsigned char [algorithm](#)
- unsigned char [digest_type](#)
- unsigned int [keytag](#)
- unsigned char * [digest](#)
- size_t [diglen](#)
- unsigned char [validated](#)

This is for the cache.

- [dnskey_t](#) ** [signkeys](#)

The DNSKEY(s) that were used to sign the RRSIG guaranteeing this DS.

4.25.1 Detailed Description

Our own internal representation of certain RRs.

Definition at line 94 of file dns.h.

4.25.2 Field Documentation

4.25.2.1 unsigned char ds::algorithm

Definition at line 96 of file dns.h.

Referenced by [_add_ds_entry\(\)](#), [_deserialize_ds_record_cb\(\)](#), [_dump_ds_record_cb\(\)](#), [_get_ds_by_dnskey\(\)](#), and [_serialize_ds_record_cb\(\)](#).

4.25.2.2 unsigned char* ds::digest

Definition at line 99 of file dns.h.

Referenced by [_add_ds_entry\(\)](#), [_deserialize_ds_record_cb\(\)](#), [_destroy_ds\(\)](#), [_ds_comparator\(\)](#), [_dump_ds_record_cb\(\)](#), [_get_ds_by_dnskey\(\)](#), and [_serialize_ds_record_cb\(\)](#).

4.25.2.3 unsigned char ds::digest_type

Definition at line 97 of file dns.h.

Referenced by [_add_ds_entry\(\)](#), [_deserialize_ds_record_cb\(\)](#), [_dump_ds_record_cb\(\)](#), [_get_ds_by_dnskey\(\)](#), and [_serialize_ds_record_cb\(\)](#).

4.25.2.4 size_t ds::diglen

Definition at line 100 of file dns.h.

Referenced by [_add_ds_entry\(\)](#), [_deserialize_ds_record_cb\(\)](#), [_ds_comparator\(\)](#), [_dump_ds_record_cb\(\)](#), [_get_ds_by_dnskey\(\)](#), and [_serialize_ds_record_cb\(\)](#).

4.25.2.5 unsigned int ds::keytag

Definition at line 98 of file dns.h.

Referenced by `_add_ds_entry()`, `_deserialize_ds_record_cb()`, `_ds_comparator()`, `_dump_ds_record_cb()`, `_get_ds_by_dnskey()`, and `_serialize_ds_record_cb()`.

4.25.2.6 char* ds::label

Definition at line 95 of file dns.h.

Referenced by `_add_ds_entry()`, `_deserialize_ds_record_cb()`, `_destroy_ds()`, `_ds_comparator()`, `_dump_ds_record_cb()`, `_get_ds_by_dnskey()`, and `_serialize_ds_record_cb()`.

4.25.2.7 dnskey_t** ds::signkeys

The DNSKEY(s) that were used to sign the RRSIG guaranteeing this DS.

Definition at line 102 of file dns.h.

Referenced by `_destroy_ds()`, and `_dump_ds_record_cb()`.

4.25.2.8 unsigned char ds::validated

This is for the cache.

Definition at line 101 of file dns.h.

Referenced by `_deserialize_ds_record_cb()`, `_dump_ds_record_cb()`, and `_serialize_ds_record_cb()`.

4.26 ED25519_KEY Struct Reference

```
#include <dcrypto.h>
```

Data Fields

- [ed25519_secret_key](#) private_key
- [ed25519_public_key](#) public_key

4.26.1 Detailed Description

Definition at line 26 of file `dcrypto.h`.

4.26.2 Field Documentation

4.26.2.1 `ed25519_secret_key` `ED25519_KEY::private_key`

Definition at line 27 of file `dcrypto.h`.

Referenced by `_deserialize_ed25519_privkey()`, `_ed25519_sign_data()`, `_generate_ed25519_keypair()`, and `_load_ed25519_privkey()`.

4.26.2.2 `ed25519_public_key` `ED25519_KEY::public_key`

Definition at line 28 of file `dcrypto.h`.

Referenced by `_deserialize_ed25519_privkey()`, `_deserialize_ed25519_pubkey()`, `_ed25519_sign_data()`, `_ed25519_verify_sig()`, `_generate_ed25519_keypair()`, `_load_ed25519_privkey()`, and `_verify_dx_certificate()`.

4.27 encrypt_ctx Struct Reference

Data Fields

- EC_GROUP * [encryption_group](#)
- EVP_MD const * [ecies_envelope_evp](#)

4.27.1 Detailed Description

Definition at line 11 of file encrypt.c.

4.27.2 Field Documentation

4.27.2.1 EVP_MD const* encrypt_ctx::ecies_envelope_evp

Definition at line 13 of file encrypt.c.

4.27.2.2 EC_GROUP* encrypt_ctx::encryption_group

Definition at line 12 of file encrypt.c.

Referenced by `encrypt_ctx_free()`, and `encrypt_keypair_generate()`.

4.28 encrypt_keypair_t Struct Reference

```
#include <encrypt.h>
```

Data Fields

- char [unused](#) [1]

4.28.1 Detailed Description

Definition at line 8 of file encrypt.h.

4.28.2 Field Documentation

4.28.2.1 char encrypt_keypair_t::unused[1]

Definition at line 8 of file encrypt.h.

4.29 `err_desc_t` Struct Reference

```
#include <error.h>
```

Data Fields

- unsigned int `errcode`
- const char * `errmsg`

4.29.1 Detailed Description

Definition at line 59 of file `error.h`.

4.29.2 Field Documentation

4.29.2.1 unsigned int `err_desc_t::errcode`

Definition at line 60 of file `error.h`.

4.29.2.2 const char* `err_desc_t::errmsg`

Definition at line 61 of file `error.h`.

Referenced by `get_error_string()`.

4.30 ge25519_niels_t Struct Reference

```
#include <ed25519-donna.h>
```

Data Fields

- [bignum25519 ysubx](#)
- [bignum25519 xaddy](#)
- [bignum25519 t2d](#)

4.30.1 Detailed Description

Definition at line 89 of file ed25519-donna.h.

4.30.2 Field Documentation

4.30.2.1 bignum25519 ge25519_niels_t::t2d

Definition at line 90 of file ed25519-donna.h.

4.30.2.2 bignum25519 ge25519_niels_t::xaddy

Definition at line 90 of file ed25519-donna.h.

4.30.2.3 bignum25519 ge25519_niels_t::ysubx

Definition at line 90 of file ed25519-donna.h.

4.31 ge25519_p1p1_t Struct Reference

```
#include <ed25519-donna.h>
```

Data Fields

- [bignum25519 x](#)
- [bignum25519 y](#)
- [bignum25519 z](#)
- [bignum25519 t](#)

4.31.1 Detailed Description

Definition at line 85 of file ed25519-donna.h.

4.31.2 Field Documentation

4.31.2.1 bignum25519 ge25519_p1p1_t::t

Definition at line 86 of file ed25519-donna.h.

4.31.2.2 bignum25519 ge25519_p1p1_t::x

Definition at line 86 of file ed25519-donna.h.

4.31.2.3 bignum25519 ge25519_p1p1_t::y

Definition at line 86 of file ed25519-donna.h.

4.31.2.4 bignum25519 ge25519_p1p1_t::z

Definition at line 86 of file ed25519-donna.h.

4.32 ge25519_pniels_t Struct Reference

```
#include <ed25519-donna.h>
```

Data Fields

- [bignum25519 ysubx](#)
- [bignum25519 xaddy](#)
- [bignum25519 z](#)
- [bignum25519 t2d](#)

4.32.1 Detailed Description

Definition at line 93 of file ed25519-donna.h.

4.32.2 Field Documentation

4.32.2.1 bignum25519 ge25519_pniels_t::t2d

Definition at line 94 of file ed25519-donna.h.

4.32.2.2 bignum25519 ge25519_pniels_t::xaddy

Definition at line 94 of file ed25519-donna.h.

4.32.2.3 bignum25519 ge25519_pniels_t::ysubx

Definition at line 94 of file ed25519-donna.h.

4.32.2.4 bignum25519 ge25519_pniels_t::z

Definition at line 94 of file ed25519-donna.h.

4.33 ge25519_t Struct Reference

```
#include <ed25519-donna.h>
```

Data Fields

- [bignum25519 x](#)
- [bignum25519 y](#)
- [bignum25519 z](#)
- [bignum25519 t](#)

4.33.1 Detailed Description

Definition at line 81 of file ed25519-donna.h.

4.33.2 Field Documentation

4.33.2.1 bignum25519 ge25519_t::t

Definition at line 82 of file ed25519-donna.h.

4.33.2.2 bignum25519 ge25519_t::x

Definition at line 82 of file ed25519-donna.h.

4.33.2.3 bignum25519 ge25519_t::y

Definition at line 82 of file ed25519-donna.h.

Referenced by `curved25519_scalarmult_basepoint()`.

4.33.2.4 bignum25519 ge25519_t::z

Definition at line 82 of file ed25519-donna.h.

Referenced by `curved25519_scalarmult_basepoint()`.

4.34 ge_cached Struct Reference

Data Fields

- [fe YplusX](#)
- [fe YminusX](#)
- [fe Z](#)
- [fe T2d](#)

4.34.1 Detailed Description

Definition at line 1703 of file ed25519-ref10.c.

4.34.2 Field Documentation

4.34.2.1 fe ge_cached::T2d

Definition at line 1707 of file ed25519-ref10.c.

4.34.2.2 fe ge_cached::YminusX

Definition at line 1705 of file ed25519-ref10.c.

4.34.2.3 fe ge_cached::YplusX

Definition at line 1704 of file ed25519-ref10.c.

4.34.2.4 fe ge_cached::Z

Definition at line 1706 of file ed25519-ref10.c.

4.35 ge_p1p1 Struct Reference

Data Fields

- [fe X](#)
- [fe Y](#)
- [fe Z](#)
- [fe T](#)

4.35.1 Detailed Description

Definition at line 1690 of file ed25519-ref10.c.

4.35.2 Field Documentation

4.35.2.1 fe ge_p1p1::T

Definition at line 1694 of file ed25519-ref10.c.

4.35.2.2 fe ge_p1p1::X

Definition at line 1691 of file ed25519-ref10.c.

4.35.2.3 fe ge_p1p1::Y

Definition at line 1692 of file ed25519-ref10.c.

4.35.2.4 fe ge_p1p1::Z

Definition at line 1693 of file ed25519-ref10.c.

4.36 ge_p2 Struct Reference

Data Fields

- [fe X](#)
- [fe Y](#)
- [fe Z](#)

4.36.1 Detailed Description

Definition at line 1677 of file ed25519-ref10.c.

4.36.2 Field Documentation

4.36.2.1 fe ge_p2::X

Definition at line 1678 of file ed25519-ref10.c.

4.36.2.2 fe ge_p2::Y

Definition at line 1679 of file ed25519-ref10.c.

4.36.2.3 fe ge_p2::Z

Definition at line 1680 of file ed25519-ref10.c.

4.37 ge_p3 Struct Reference

Data Fields

- [fe X](#)
- [fe Y](#)
- [fe Z](#)
- [fe T](#)

4.37.1 Detailed Description

Definition at line 1683 of file ed25519-ref10.c.

4.37.2 Field Documentation

4.37.2.1 fe ge_p3::T

Definition at line 1687 of file ed25519-ref10.c.

4.37.2.2 fe ge_p3::X

Definition at line 1684 of file ed25519-ref10.c.

4.37.2.3 fe ge_p3::Y

Definition at line 1685 of file ed25519-ref10.c.

4.37.2.4 fe ge_p3::Z

Definition at line 1686 of file ed25519-ref10.c.

4.38 ge_precomp Struct Reference

Data Fields

- [fe yplusx](#)
- [fe yminusx](#)
- [fe xy2d](#)

4.38.1 Detailed Description

Definition at line 1697 of file ed25519-ref10.c.

4.38.2 Field Documentation

4.38.2.1 fe ge_precomp::xy2d

Definition at line 1700 of file ed25519-ref10.c.

4.38.2.2 fe ge_precomp::yminusx

Definition at line 1699 of file ed25519-ref10.c.

4.38.2.3 fe ge_precomp::yplusx

Definition at line 1698 of file ed25519-ref10.c.

4.39 generated_data_t Struct Reference

Data Fields

- unsigned char [pk](#) [32]
- unsigned char [sig](#) [64]
- int [valid](#)

4.39.1 Detailed Description

Definition at line 121 of file fuzz-ed25519.c.

4.39.2 Field Documentation

4.39.2.1 unsigned char generated_data_t::pk[32]

Definition at line 122 of file fuzz-ed25519.c.

Referenced by `main()`.

4.39.2.2 unsigned char generated_data_t::sig[64]

Definition at line 123 of file fuzz-ed25519.c.

Referenced by `main()`.

4.39.2.3 int generated_data_t::valid

Definition at line 124 of file fuzz-ed25519.c.

Referenced by `main()`.

4.40 http_content_t Struct Reference

```
#include <http.h>
```

Data Fields

- [stringer_t](#) * location
- [stringer_t](#) * resource
- [stringer_t](#) * type
- struct [http_content_t](#) * next

4.40.1 Detailed Description

Definition at line 41 of file http.h.

4.40.2 Field Documentation

4.40.2.1 stringer_t* http_content_t::location

Definition at line 42 of file http.h.

Referenced by [http_free_content\(\)](#).

4.40.2.2 struct http_content_t* http_content_t::next [read]

Definition at line 43 of file http.h.

4.40.2.3 stringer_t* http_content_t::resource

Definition at line 42 of file http.h.

Referenced by [http_free_content\(\)](#), [http_page_get\(\)](#), [http_response\(\)](#), [register_print_message\(\)](#), [register_print_step1\(\)](#), [register_print_step2\(\)](#), and [register_print_step3\(\)](#).

4.40.2.4 stringer_t* http_content_t::type

Definition at line 42 of file http.h.

Referenced by [contact_print_form\(\)](#), [contact_print_message\(\)](#), [http_free_content\(\)](#), [http_get_static\(\)](#), [http_get_template\(\)](#), [http_response\(\)](#), [portal_print_login\(\)](#), [register_print_message\(\)](#), [register_print_step1\(\)](#), [register_print_step2\(\)](#), [register_print_step3\(\)](#), [teacher_print_form\(\)](#), and [teacher_print_message\(\)](#).

4.41 http_data_t Struct Reference

```
#include <http.h>
```

Data Fields

- [HTTP_DATA](#) source
- [stringer_t](#) * name
- [stringer_t](#) * value

4.41.1 Detailed Description

Definition at line 36 of file http.h.

4.41.2 Field Documentation

4.41.2.1 stringer_t* http_data_t::name

Definition at line 38 of file http.h.

Referenced by [http_data_free\(\)](#), [http_data_get\(\)](#), [http_data_header_parse_line\(\)](#), [http_data_value_parse\(\)](#), [http_parse_header\(\)](#), and [portal_upload\(\)](#).

4.41.2.2 HTTP_DATA http_data_t::source

Definition at line 37 of file http.h.

Referenced by [http_data_get\(\)](#), [http_data_header_parse_line\(\)](#), and [http_data_value_parse\(\)](#).

4.41.2.3 stringer_t * http_data_t::value

Definition at line 38 of file http.h.

Referenced by [contact_business\(\)](#), [contact_print_form\(\)](#), [http_body\(\)](#), [http_data_free\(\)](#), [http_data_header_parse_line\(\)](#), [http_data_value_parse\(\)](#), [http_parse_header\(\)](#), [http_response_allow_cross\(\)](#), [multipart_get_boundary\(\)](#), [portal_upload\(\)](#), [register_business_step1\(\)](#), [register_business_step2\(\)](#), [register_process\(\)](#), and [teacher_process\(\)](#).

4.42 http_page_t Struct Reference

```
#include <http.h>
```

Data Fields

- xmlDocPtr [doc_obj](#)
- [http_content_t](#) * [content](#)
- xmlParserCtxtPtr [doc_ctx](#)
- xmlXPathContextPtr [xpath_ctx](#)

4.42.1 Detailed Description

Definition at line 46 of file `http.h`.

4.42.2 Field Documentation

4.42.2.1 http_content_t* http_page_t::content

Definition at line 48 of file `http.h`.

Referenced by `contact_print_form()`, `contact_print_message()`, `http_page_get()`, `portal_print_login()`, `teacher_print_form()`, and `teacher_print_message()`.

4.42.2.2 xmlParserCtxtPtr http_page_t::doc_ctx

Definition at line 49 of file `http.h`.

Referenced by `http_page_free()`, and `http_page_get()`.

4.42.2.3 xmlDocPtr http_page_t::doc_obj

Definition at line 47 of file `http.h`.

Referenced by `contact_print_form()`, `contact_print_message()`, `http_page_free()`, `http_page_get()`, `portal_print_login()`, `teacher_print_form()`, and `teacher_print_message()`.

4.42.2.4 xmlXPathContextPtr http_page_t::xpath_ctx

Definition at line 50 of file `http.h`.

Referenced by `contact_business_add_error()`, `contact_print_form()`, `contact_print_message()`, `http_page_free()`, `http_page_get()`, `portal_print_login()`, `teacher_add_error()`, `teacher_print_form()`, and `teacher_print_message()`.

4.43 imap_fetch_dataitems_t Struct Reference

```
#include <imap.h>
```

Data Fields

- [int_t uid](#)
- [int_t flags](#)
- [int_t internaldate](#)
- [int_t envelope](#)
- [int_t bodystructure](#)
- [int_t rfc822](#)
- [int_t rfc822_header](#)
- [int_t rfc822_size](#)
- [int_t rfc822_text](#)
- [int_t body](#)
- [array_t * peek](#)
- [array_t * peek_partial](#)
- [array_t * normal](#)
- [array_t * normal_partial](#)

4.43.1 Detailed Description

Definition at line 18 of file `imap.h`.

4.43.2 Field Documentation

4.43.2.1 `int_t imap_fetch_dataitems_t::body`

Definition at line 19 of file `imap.h`.

Referenced by `imap_fetch_message()`, and `imap_parse_dataitems()`.

4.43.2.2 `int_t imap_fetch_dataitems_t::bodystructure`

Definition at line 19 of file `imap.h`.

Referenced by `imap_fetch_message()`, and `imap_parse_dataitems()`.

4.43.2.3 `int_t imap_fetch_dataitems_t::envelope`

Definition at line 19 of file `imap.h`.

Referenced by `imap_fetch_message()`, and `imap_parse_dataitems()`.

4.43.2.4 `int_t imap_fetch_dataitems_t::flags`

Definition at line 19 of file `imap.h`.

Referenced by `imap_fetch_message()`, and `imap_parse_dataitems()`.

4.43.2.5 int_t imap_fetch_dataitems_t::internaldate

Definition at line 19 of file imap.h.

Referenced by imap_fetch_message(), and imap_parse_dataitems().

4.43.2.6 array_t * imap_fetch_dataitems_t::normal

Definition at line 20 of file imap.h.

Referenced by imap_fetch(), imap_fetch_free_items(), imap_fetch_message(), and imap_parse_dataitems().

4.43.2.7 array_t * imap_fetch_dataitems_t::normal_partial

Definition at line 20 of file imap.h.

Referenced by imap_fetch_free_items(), imap_fetch_message(), and imap_parse_dataitems().

4.43.2.8 array_t* imap_fetch_dataitems_t::peek

Definition at line 20 of file imap.h.

Referenced by imap_fetch_free_items(), imap_fetch_message(), and imap_parse_dataitems().

4.43.2.9 array_t * imap_fetch_dataitems_t::peek_partial

Definition at line 20 of file imap.h.

Referenced by imap_fetch_free_items(), imap_fetch_message(), and imap_parse_dataitems().

4.43.2.10 int_t imap_fetch_dataitems_t::rfc822

Definition at line 19 of file imap.h.

Referenced by imap_fetch(), imap_fetch_message(), and imap_parse_dataitems().

4.43.2.11 int_t imap_fetch_dataitems_t::rfc822_header

Definition at line 19 of file imap.h.

Referenced by imap_fetch(), imap_fetch_message(), and imap_parse_dataitems().

4.43.2.12 int_t imap_fetch_dataitems_t::rfc822_size

Definition at line 19 of file imap.h.

Referenced by imap_fetch_message(), and imap_parse_dataitems().

4.43.2.13 int_t imap_fetch_dataitems_t::rfc822_text

Definition at line 19 of file imap.h.

Referenced by imap_fetch(), imap_fetch_message(), and imap_parse_dataitems().

4.43.2.14 `int_t imap_fetch_dataitems_t::uid`

Definition at line 19 of file `imap.h`.

Referenced by `imap_fetch_message()`, and `imap_parse_dataitems()`.

4.44 imap_fetch_response_t Struct Reference

```
#include <imap.h>
```

Data Fields

- [stringer_t](#) * key
- [stringer_t](#) * value
- struct [imap_fetch_response_t](#) * next

4.44.1 Detailed Description

Definition at line 23 of file `imap.h`.

4.44.2 Field Documentation

4.44.2.1 `stringer_t*` `imap_fetch_response_t::key`

Definition at line 24 of file `imap.h`.

Referenced by `imap_fetch()`, `imap_fetch_response_add()`, `imap_fetch_response_free()`, `smtp_check_greylist()`, and `smtp_store_spamsig()`.

4.44.2.2 `struct imap_fetch_response_t*` `imap_fetch_response_t::next` `[read]`

Definition at line 25 of file `imap.h`.

Referenced by `imap_fetch()`, `imap_fetch_response_add()`, and `imap_fetch_response_free()`.

4.44.2.3 `stringer_t` * `imap_fetch_response_t::value`

Definition at line 24 of file `imap.h`.

Referenced by `imap_fetch()`, `imap_fetch_response_add()`, `imap_fetch_response_free()`, `imap_parse_address()`, and `smtp_check_greylist()`.

4.45 imap_folder_status_t Struct Reference

```
#include <imap.h>
```

Data Fields

- uint64_t [foldernum](#)
- uint64_t [recent](#)
- uint64_t [unseen](#)
- uint64_t [uidnext](#)
- uint64_t [messages](#)
- uint64_t [first](#)

4.45.1 Detailed Description

Definition at line 14 of file `imap.h`.

4.45.2 Field Documentation

4.45.2.1 uint64_t imap_folder_status_t::first

Definition at line 15 of file `imap.h`.

Referenced by `imap_examine()`, `imap_folder_status()`, and `imap_select()`.

4.45.2.2 uint64_t imap_folder_status_t::foldernum

Definition at line 15 of file `imap.h`.

Referenced by `imap_examine()`, `imap_folder_status()`, `imap_select()`, and `imap_status()`.

4.45.2.3 uint64_t imap_folder_status_t::messages

Definition at line 15 of file `imap.h`.

Referenced by `imap_examine()`, `imap_folder_status()`, `imap_select()`, and `imap_status()`.

4.45.2.4 uint64_t imap_folder_status_t::recent

Definition at line 15 of file `imap.h`.

Referenced by `imap_examine()`, `imap_folder_status()`, `imap_select()`, and `imap_status()`.

4.45.2.5 uint64_t imap_folder_status_t::uidnext

Definition at line 15 of file `imap.h`.

Referenced by `imap_examine()`, `imap_folder_status()`, `imap_select()`, and `imap_status()`.

4.45.2.6 uint64_t imap_folder_status_t::unseen

Definition at line 15 of file `imap.h`.

Referenced by `imap_folder_status()`, and `imap_status()`.

4.46 ip_t Struct Reference

```
#include <host.h>
```

Data Fields

- sa_family_t [family](#)
- union {
 - struct in_addr [ip4](#)
 - struct in6_addr [ip6](#)
 - void * [ip](#)

4.46.1 Detailed Description

Definition at line 68 of file host.h.

4.46.2 Field Documentation

4.46.2.1 union { ... }

4.46.2.2 sa_family_t ip_t::family

Definition at line 69 of file host.h.

Referenced by `client_connect()`, `ip_addr_eq()`, `ip_addr_st()`, `ip_family()`, `ip_matches_subnet()`, `ip_octet()`, `ip_presentation()`, `ip_reversed()`, `ip_segment()`, `ip_standard()`, `ip_subnet()`, `ip_subnet_st()`, `ip_word()`, `net_trigger()`, `spf_check()`, and `tcp_addr_ip()`.

4.46.2.3 void* ip_t::ip

Definition at line 73 of file host.h.

Referenced by `con_addr()`, and `ip_matches_subnet()`.

4.46.2.4 struct in_addr ip_t::ip4 [read]

Definition at line 71 of file host.h.

Referenced by `client_connect()`, `ip_addr_eq()`, `ip_addr_st()`, `ip_octet()`, `ip_presentation()`, `ip_reversed()`, `ip_segment()`, `ip_standard()`, `ip_subnet()`, `ip_word()`, `net_trigger()`, `spf_check()`, and `tcp_addr_ip()`.

4.46.2.5 struct in6_addr ip_t::ip6 [read]

Definition at line 72 of file host.h.

Referenced by `client_connect()`, `ip_addr_eq()`, `ip_addr_st()`, `ip_localhost()`, `ip_octet()`, `ip_presentation()`, `ip_private()`, `ip_reversed()`, `ip_segment()`, `ip_standard()`, `ip_subnet()`, `ip_word()`, `net_trigger()`, `spf_check()`, and `tcp_addr_ip()`.

4.47 key_pair_t Struct Reference

```
#include <meta.h>
```

Data Fields

- [stringer_t * public](#)
- [stringer_t * private](#)

4.47.1 Detailed Description

Definition at line 11 of file meta.h.

4.47.2 Field Documentation

4.47.2.1 stringer_t* key_pair_t::private

Definition at line 13 of file meta.h.

Referenced by meta_crypto_keys_create(), meta_data_fetch_keys(), meta_data_insert_keys(), and meta_update_keys().

4.47.2.2 stringer_t* key_pair_t::public

Definition at line 12 of file meta.h.

Referenced by meta_crypto_keys_create(), meta_data_fetch_keys(), meta_data_insert_keys(), and meta_update_keys().

4.48 log_level_t Struct Reference

```
#include <dime_ctx.h>
```

Data Fields

- [log_code_t](#) code
- char const * [name](#)

4.48.1 Detailed Description

Definition at line 16 of file dime_ctx.h.

4.48.2 Field Documentation

4.48.2.1 log_code_t log_level_t::code

Definition at line 17 of file dime_ctx.h.

4.48.2.2 char const* log_level_t::name

Definition at line 18 of file dime_ctx.h.

4.49 magma_keys_t Struct Reference

```
#include <global.h>
```

Data Fields

- void * store
- multi_t norm
- chr_t * name
- chr_t * description
- bool_t file
- bool_t database
- bool_t overwrite
- bool_t set
- bool_t required

4.49.1 Detailed Description

Definition at line 11 of file global.h.

4.49.2 Field Documentation

4.49.2.1 bool_t magma_keys_t::database

Definition at line 17 of file global.h.

Referenced by config_load_database_settings().

4.49.2.2 chr_t* magma_keys_t::description

Definition at line 15 of file global.h.

4.49.2.3 bool_t magma_keys_t::file

Definition at line 16 of file global.h.

Referenced by config_load_cmdline_settings(), and config_load_file_settings().

4.49.2.4 chr_t* magma_keys_t::name

Definition at line 14 of file global.h.

Referenced by config_load_cmdline_settings(), config_load_database_settings(), config_load_file_settings(), config_output_value(), and config_value_set().

4.49.2.5 multi_t magma_keys_t::norm

Definition at line 13 of file global.h.

Referenced by config_free(), config_output_help(), config_output_value(), and config_value_set().

4.49.2.6 bool_t magma_keys_t::overwrite

Definition at line 18 of file global.h.

Referenced by config_load_database_settings().

4.49.2.7 bool_t magma_keys_t::required

Definition at line 20 of file global.h.

Referenced by config_load_cmdline_settings(), config_load_database_settings(), config_load_file_settings(), and config_output_help().

4.49.2.8 bool_t magma_keys_t::set

Definition at line 19 of file global.h.

Referenced by config_load_cmdline_settings(), config_load_database_settings(), config_load_file_settings(), and config_validate_settings().

4.49.2.9 void* magma_keys_t::store

Definition at line 12 of file global.h.

Referenced by config_output_value(), and config_value_set().

4.50 magma_t Struct Reference

```
#include <global.h>
```

Data Fields

- struct {
 - bool_t output_config
 - bool_t output_resource_limits
 - chr_t file [MAGMA_FILEPATH_MAX+1]
 } config
- struct {
 - stringer_t * contact
 - stringer_t * abuse
 } admin
- struct {
 - uint64_t number
 - chr_t name [MAGMA_HOSTNAME_MAX+1]
 } host
- struct {
 - chr_t * file
 - bool_t unload
 } library
- struct {
 - bool_t daemonize
 - char * root_directory
 - char * impersonate_user
 - bool_t increase_resource_limits
 - uint32_t thread_stack_size
 - uint32_t worker_threads
 - uint32_t network_buffer
 - bool_t enable_core_dumps
 - uint64_t core_dump_size_limit
 - stringer_t * domain
 } system
- struct {
 - struct {
 - bool_t enable
 - uint64_t length
 } memory
 - uint32_t minimum_password_length
 - stringer_t * salt
 - stringer_t * links
 - stringer_t * sessions
 } secure
- struct {
 - bool_t file
 - chr_t * path
 } output

- struct {
 - bool_t imap
 - bool_t http
 - bool_t content
 - bool_t file
 - bool_t line
 - bool_t time
 - bool_t stack
 - bool_t function
 } log
- struct {
 - chr_t * tank
 - stringer_t * active
 - stringer_t * root
 } storage
- struct {
 - uint32_t relay_limit
 - uint32_t recipient_limit
 - uint32_t address_length_limit
 - uint32_t helo_length_limit
 - uint32_t wrap_line_length
 - uint64_t message_length_limit
 - struct {
 - uint32_t count
 - stringer_t * domain [MAGMA_BLACKLIST_INSTANCES]
 } blacklists
 - stringer_t * bypass_addr
 - inx_t * bypass_subnets
 } smtp
- struct {
 - bool_t close
 - bool_t allow_cross_domain
 - chr_t * fonts
 - chr_t * pages
 - chr_t * templates
 - uint32_t session_timeout
 } http
- struct {
 - relay_t * host [MAGMA_RELAY_INSTANCES]
 - struct {
 - uint32_t premium
 - uint32_t standard
 } count
 - uint32_t timeout
 } relay
- struct {
 - struct {
 - bool_t indent
 - bool_t safeguard
 } portal
 - struct {
 - stringer_t * sender

```

    } contact
    bool_t statistics
    bool_t registration
    stringer_t * tls_redirect
} web

• struct {
    bool_t enabled
    chr_t * domain
    chr_t * selector
    stringer_t * key
} dkin

• struct {
    stringer_t * key
    stringer_t * signet
} dime

• struct {
    struct {
        uint32_t seed_length
        bool_t dhparams_rotate
        bool_t dhparams_large_keys
    } cryptography
    struct {
        chr_t * host
        chr_t * user
        chr_t * password
        chr_t * schema
        uint32_t port
        chr_t * socket_path
        struct {
            uint32_t timeout
            uint32_t connections
        } pool
    } database
    struct {
        bool_t available
        char * signatures
    } virus
    struct {
        chr_t * path
        stringer_t * cache
    } location
    struct {
        cache_t * host [MAGMA_CACHE_INSTANCES]
        struct {
            uint32_t timeout
            uint32_t connections
        } pool
        uint32_t retry
        uint32_t timeout
    } cache
    struct {
        struct {
            uint32_t timeout
            uint32_t connections

```

```

    } pool
  } spf
} iface

```

- `chr_t * spool`
- `int_t page_length`
- `uint32_t init`
- `pthread_rwlock_t lock`
- `server_t * servers` [MAGMA_SERVER_INSTANCES]

4.50.1 Detailed Description

Definition at line 23 of file global.h.

4.50.2 Field Documentation

4.50.2.1 `stringer_t* magma_t::abuse`

Definition at line 35 of file global.h.

Referenced by `config_validate_settings()`, `contact_business()`, `http_response()`, `smtp_accept_message()`, and `smtp_rcpt_to()`.

4.50.2.2 `stringer_t* magma_t::active`

Definition at line 96 of file global.h.

Referenced by `imap_append_message()`, `mail_create_directory()`, `mail_db_insert_duplicate_message()`, `mail_db_insert_message()`, and `mail_message_path()`.

4.50.2.3 `uint32_t magma_t::address_length_limit`

Definition at line 103 of file global.h.

Referenced by `smtp_parse_mail_from_path()`, and `smtp_parse_rcpt_to()`.

4.50.2.4 `struct { ... } magma_t::admin`

Referenced by `config_validate_settings()`, `contact_business()`, `http_response()`, `register_business_step2()`, `smtp_accept_message()`, and `smtp_rcpt_to()`.

4.50.2.5 `bool_t magma_t::allow_cross_domain`

Definition at line 120 of file global.h.

4.50.2.6 `bool_t magma_t::available`

Definition at line 184 of file global.h.

4.50.2.7 `struct { ... } magma_t::blacklists`

Referenced by `config_free()`, `config_output_value()`, `config_output_value_generic()`, `config_value_set()`, and `smtp_check_rbl()`.

4.50.2.8 `stringer_t* magma_t::bypass_addr`

Definition at line 114 of file global.h.

4.50.2.9 `inx_t* magma_t::bypass_subnets`

Definition at line 115 of file global.h.

Referenced by `smtp_add_bypass_entry()`, and `smtp_bypass_check()`.

4.50.2.10 `struct { ... } magma_t::cache`

4.50.2.11 `stringer_t* magma_t::cache`

Definition at line 190 of file global.h.

Referenced by `cache_alloc()`, `cache_free()`, `cache_output_settings()`, `cache_start()`, `cache_stop()`, `cache_validate()`, and `config_validate_settings()`.

4.50.2.12 `bool_t magma_t::close`

Definition at line 119 of file global.h.

4.50.2.13 `struct { ... } magma_t::config`

Referenced by `args_parse()`, `config_load_file_settings()`, `config_output_settings()`, `config_validate_settings()`, `lib_load()`, and `system_init_resource_limits()`.

4.50.2.14 `uint32_t magma_t::connections`

Definition at line 179 of file global.h.

4.50.2.15 `struct { ... } magma_t::contact`

4.50.2.16 `stringer_t* magma_t::contact`

Definition at line 34 of file global.h.

Referenced by `config_validate_settings()`, `contact_business()`, `http_response()`, `register_business_step2()`, `smtp_accept_message()`, and `smtp_rcpt_to()`.

4.50.2.17 `bool_t magma_t::content`

Definition at line 85 of file global.h.

Referenced by `http_content_load_fonts()`, and `http_load_file()`.

4.50.2.18 `uint64_t magma_t::core_dump_size_limit`

Definition at line 58 of file global.h.

Referenced by `system_init_core_dumps()`.

4.50.2.19 struct { ... } magma_t::count**4.50.2.20 uint32_t magma_t::count**

Definition at line 110 of file global.h.

Referenced by relay_counter(), and smtp_client_connect().

4.50.2.21 struct { ... } magma_t::cryptography

Referenced by dh_params_generate(), rand_start(), rand_thread_start(), and tls_server_create().

4.50.2.22 bool_t magma_t::daemonize

Definition at line 49 of file global.h.

Referenced by color_supported(), config_validate_settings(), log_rotate(), log_start(), and system_fork_daemon().

4.50.2.23 struct { ... } magma_t::database

Referenced by sql_open(), sql_start(), sql_stop(), stmt_start(), and stmt_stop().

4.50.2.24 bool_t magma_t::dhparams_large_keys

Definition at line 166 of file global.h.

4.50.2.25 bool_t magma_t::dhparams_rotate

Definition at line 165 of file global.h.

4.50.2.26 struct { ... } magma_t::dime

Referenced by config_validate_settings(), and prime_start().

4.50.2.27 struct { ... } magma_t::dkim

Referenced by config_validate_settings(), dkim_signature_create(), dkim_start(), mail_add_forward_headers(), and mail_add_outbound_headers().

4.50.2.28 chr_t* magma_t::domain

Definition at line 151 of file global.h.

4.50.2.29 stringer_t* magma_t::domain[MAGMA_BLACKLIST_INSTANCES]

Definition at line 60 of file global.h.

Referenced by auth_sanitize_username(), config_validate_settings(), dkim_signature_create(), register_business_step2(), register_data_insert_user(), smtp_bounce(), and smtp_client_send_helo().

4.50.2.30 `bool_t magma_t::enable`

Definition at line 65 of file global.h.

4.50.2.31 `bool_t magma_t::enable_core_dumps`

Definition at line 57 of file global.h.

Referenced by `system_init_core_dumps()`.

4.50.2.32 `bool_t magma_t::enabled`

Definition at line 150 of file global.h.

Referenced by `config_validate_settings()`, `dkim_signature_create()`, `dkim_start()`, `mail_add_forward_headers()`, and `mail_add_outbound_headers()`.

4.50.2.33 `bool_t magma_t::file`

Definition at line 77 of file global.h.

4.50.2.34 `chr_t* magma_t::file`

Definition at line 44 of file global.h.

4.50.2.35 `chr_t magma_t::file[MAGMA_FILEPATH_MAX+1]`

Definition at line 30 of file global.h.

Referenced by `args_parse()`, `config_load_file_settings()`, `config_output_settings()`, `config_validate_settings()`, `lib_load()`, `log_internal()`, `log_rotate()`, and `log_start()`.

4.50.2.36 `chr_t* magma_t::fonts`

Definition at line 121 of file global.h.

4.50.2.37 `bool_t magma_t::function`

Definition at line 91 of file global.h.

Referenced by `log_internal()`.

4.50.2.38 `uint32_t magma_t::helo_length_limit`

Definition at line 104 of file global.h.

Referenced by `smtp_parse_helo_domain()`.

4.50.2.39 `cache_t* magma_t::host[MAGMA_CACHE_INSTANCES]`

Definition at line 194 of file global.h.

4.50.2.40 chr_t* magma_t::host

Definition at line 170 of file global.h.

4.50.2.41 relay_t* magma_t::host[MAGMA_RELAY_INSTANCES]

Definition at line 128 of file global.h.

4.50.2.42 struct { ... } magma_t::host

Referenced by config_fetch_host_number(), config_fetch_settings(), config_load_database_settings(), config_output_settings(), process_start(), relay_alloc(), relay_counter(), relay_free(), relay_output_settings(), relay_validate(), servers_network_start(), sess_create(), smtp_client_connect(), and smtp_client_send_helo().

4.50.2.43 struct { ... } magma_t::http**4.50.2.44 bool_t magma_t::http**

Definition at line 84 of file global.h.

Referenced by config_validate_settings(), http_body(), http_content_load_fonts(), http_content_refresh(), http_content_start(), http_data_value_parse(), http_load_file(), http_parse_header(), http_parse_method(), http_response_connection(), http_response_header(), http_response_options(), register_captcha_random_font(), and sess_get().

4.50.2.45 struct { ... } magma_t::iface

Referenced by cache_alloc(), cache_free(), cache_output_settings(), cache_start(), cache_stop(), cache_validate(), config_validate_settings(), dh_params_generate(), rand_start(), rand_thread_start(), spf_start(), spf_stop(), sql_open(), sql_start(), sql_stop(), stmt_start(), stmt_stop(), tls_server_create(), virus_check(), virus_engine_create(), virus_engine_refresh(), virus_sigs_total(), virus_start(), and virus_stop().

4.50.2.46 bool_t magma_t::imap

Definition at line 83 of file global.h.

Referenced by imap_command_parser().

4.50.2.47 char* magma_t::impersonate_user

Definition at line 51 of file global.h.

Referenced by system_init_impersonation().

4.50.2.48 bool_t magma_t::increase_resource_limits

Definition at line 52 of file global.h.

Referenced by system_init_resource_limits().

4.50.2.49 bool_t magma_t::indent

Definition at line 138 of file global.h.

4.50.2.50 uint32_t magma_t::init

Definition at line 217 of file global.h.

Referenced by process_start(), and process_stop().

4.50.2.51 stringer_t* magma_t::key

Definition at line 153 of file global.h.

Referenced by config_validate_settings(), dkim_signature_create(), dkim_start(), and prime_start().

4.50.2.52 uint64_t magma_t::length

Definition at line 66 of file global.h.

4.50.2.53 struct { ... } magma_t::library

Referenced by config_validate_settings(), lib_load(), and lib_unload().

4.50.2.54 bool_t magma_t::line

Definition at line 88 of file global.h.

Referenced by log_internal().

4.50.2.55 stringer_t* magma_t::links

Definition at line 72 of file global.h.

4.50.2.56 struct { ... } magma_t::location**4.50.2.57 pthread_rwlock_t magma_t::lock**

Definition at line 218 of file global.h.

4.50.2.58 struct { ... } magma_t::log

Referenced by http_body(), http_content_load_fonts(), http_data_value_parse(), http_load_file(), http_parse_header(), http_parse_method(), imap_command_parser(), and log_internal().

4.50.2.59 struct { ... } magma_t::memory

Referenced by mm_sec_start(), and st_realloc().

4.50.2.60 uint64_t magma_t::message_length_limit

Definition at line 106 of file global.h.

Referenced by smtp_ehlo(), and smtp_mail_from().

4.50.2.61 uint32_t magma_t::minimum_password_length

Definition at line 70 of file global.h.

Referenced by auth_login(), config_validate_settings(), register_business_step1(), register_business_validate_password(), and register_data_insert_user().

4.50.2.62 chr_t magma_t::name[MAGMA_HOSTNAME_MAX+1]

Definition at line 40 of file global.h.

Referenced by config_fetch_host_number(), config_fetch_settings(), config_output_settings(), process_start(), servers_network_start(), and smtp_client_send_helo().

4.50.2.63 uint32_t magma_t::network_buffer

Definition at line 55 of file global.h.

Referenced by con_init_network_buffer(), and net_init().

4.50.2.64 uint64_t magma_t::number

Definition at line 39 of file global.h.

Referenced by config_load_database_settings(), config_output_settings(), and sess_create().

4.50.2.65 struct { ... } magma_t::output

Referenced by config_validate_settings(), log_internal(), log_rotate(), and log_start().

4.50.2.66 bool_t magma_t::output_config

Definition at line 26 of file global.h.

Referenced by config_validate_settings().

4.50.2.67 bool_t magma_t::output_resource_limits

Definition at line 27 of file global.h.

Referenced by system_init_resource_limits().

4.50.2.68 int_t magma_t::page_length

Definition at line 214 of file global.h.

Referenced by mm_sec_start(), process_start(), st_alloc_opts(), and st_realloc().

4.50.2.69 chr_t* magma_t::pages

Definition at line 122 of file global.h.

4.50.2.70 chr_t* magma_t::password

Definition at line 172 of file global.h.

4.50.2.71 chr_t* magma_t::path

Definition at line 78 of file global.h.

Referenced by config_validate_settings(), log_rotate(), and log_start().

4.50.2.72 struct { ... } ::@42 magma_t::pool**4.50.2.73 struct { ... } ::@41 magma_t::pool****4.50.2.74 struct { ... } ::@40 magma_t::pool****4.50.2.75 uint32_t magma_t::port**

Definition at line 174 of file global.h.

4.50.2.76 struct { ... } magma_t::portal

Referenced by api_response(), jansson_flags(), json_api_dispatch(), portal_endpoint(), portal_endpoint_error(), portal_endpoint_response(), portal_process(), and portal_upload().

4.50.2.77 uint32_t magma_t::premium

Definition at line 130 of file global.h.

Referenced by relay_counter(), and smtp_client_connect().

4.50.2.78 uint32_t magma_t::recipient_limit

Definition at line 102 of file global.h.

Referenced by config_validate_settings(), and smtp_rcpt_to().

4.50.2.79 bool_t magma_t::registration

Definition at line 145 of file global.h.

Referenced by http_response().

4.50.2.80 struct { ... } magma_t::relay

Referenced by relay_alloc(), relay_counter(), relay_free(), relay_output_settings(), relay_validate(), and smtp_client_connect().

4.50.2.81 uint32_t magma_t::relay_limit

Definition at line 101 of file global.h.

Referenced by config_validate_settings(), and smtp_data().

4.50.2.82 uint32_t magma_t::retry

Definition at line 199 of file global.h.

4.50.2.83 stringer_t* magma_t::root

Definition at line 97 of file global.h.

Referenced by mail_create_directory(), and mail_message_path().

4.50.2.84 char* magma_t::root_directory

Definition at line 50 of file global.h.

Referenced by config_validate_settings(), and system_change_root_directory().

4.50.2.85 bool_t magma_t::safeguard

Definition at line 139 of file global.h.

4.50.2.86 stringer_t* magma_t::salt

Definition at line 71 of file global.h.

Referenced by auth_legacy().

4.50.2.87 chr_t* magma_t::schema

Definition at line 173 of file global.h.

4.50.2.88 struct { ... } magma_t::secure

Referenced by auth_legacy(), auth_login(), config_validate_settings(), mm_sec_start(), register_business_step1(), register_business_validate_password(), register_data_insert_user(), sess_get(), sess_token(), and st_realloc().

4.50.2.89 uint32_t magma_t::seed_length

Definition at line 164 of file global.h.

4.50.2.90 chr_t* magma_t::selector

Definition at line 152 of file global.h.

Referenced by config_validate_settings(), and dkim_signature_create().

4.50.2.91 stringer_t* magma_t::sender

Definition at line 142 of file global.h.

4.50.2.92 server_t* magma_t::servers[MAGMA_SERVER_INSTANCES]

Definition at line 219 of file global.h.

Referenced by net_listen(), net_trigger(), servers_alloc(), servers_encryption_start(), servers_encryption_stop(), servers_free(), servers_get_by_protocol(), servers_get_by_socket(), servers_get_count_using_port(), servers_network_start(), servers_network_stop(), servers_output_settings(), and servers_validate().

4.50.2.93 uint32_t magma_t::session_timeout

Definition at line 124 of file global.h.

4.50.2.94 stringer_t* magma_t::sessions

Definition at line 73 of file global.h.

Referenced by sess_get(), and sess_token().

4.50.2.95 char* magma_t::signatures

Definition at line 185 of file global.h.

4.50.2.96 stringer_t* magma_t::signet

Definition at line 158 of file global.h.

Referenced by config_validate_settings(), and prime_start().

4.50.2.97 struct { ... } magma_t::smtp

Referenced by config_free(), config_output_value(), config_output_value_generic(), config_validate_settings(), config_value_set(), mail_message_cleanup(), smtp_add_bypass_entry(), smtp_bypass_check(), smtp_check_rbl(), smtp_data(), smtp_ehlo(), smtp_mail_from(), smtp_parse_helo_domain(), smtp_parse_mail_from_path(), smtp_parse_rcpt_to(), and smtp_rcpt_to().

4.50.2.98 chr_t* magma_t::socket_path

Definition at line 175 of file global.h.

4.50.2.99 struct { ... } magma_t::spf

Referenced by spf_start(), and spf_stop().

4.50.2.100 chr_t* magma_t::spool

Definition at line 213 of file global.h.

Referenced by config_validate_settings(), spool_path(), and virus_start().

4.50.2.101 bool_t magma_t::stack

Definition at line 90 of file global.h.

Referenced by log_internal().

4.50.2.102 uint32_t magma_t::standard

Definition at line 131 of file global.h.

4.50.2.103 bool_t magma_t::statistics

Definition at line 144 of file global.h.

Referenced by http_response().

4.50.2.104 struct { ... } magma_t::storage

Referenced by imap_append_message(), mail_create_directory(), mail_db_insert_duplicate_message(), mail_db_insert_message(), mail_message_path(), and tank_start().

4.50.2.105 struct { ... } magma_t::system

Referenced by auth_sanitize_username(), color_supported(), con_init_network_buffer(), config_validate_settings(), log_rotate(), log_start(), net_init(), queue_init(), queue_shutdown(), queue_signal(), register_business_step2(), register_data_insert_user(), smtp_bounce(), smtp_client_send_helo(), system_change_root_directory(), system_fork_daemon(), system_init_core_dumps(), system_init_impersonation(), system_init_resource_limits(), and thread_launch().

4.50.2.106 chr_t* magma_t::tank

Definition at line 95 of file global.h.

Referenced by tank_start().

4.50.2.107 chr_t* magma_t::templates

Definition at line 123 of file global.h.

4.50.2.108 uint32_t magma_t::thread_stack_size

Definition at line 53 of file global.h.

Referenced by config_validate_settings(), and thread_launch().

4.50.2.109 bool_t magma_t::time

Definition at line 89 of file global.h.

Referenced by log_internal().

4.50.2.110 uint32_t magma_t::timeout

Definition at line 133 of file global.h.

Referenced by smtp_client_connect().

4.50.2.111 stringer_t* magma_t::tls_redirect

Definition at line 146 of file global.h.

Referenced by http_print_301().

4.50.2.112 `bool_t magma_t::unload`

Definition at line 45 of file global.h.

Referenced by lib_unload().

4.50.2.113 `chr_t* magma_t::user`

Definition at line 171 of file global.h.

4.50.2.114 `struct { ... } magma_t::virus`

Referenced by config_validate_settings(), virus_check(), virus_engine_create(), virus_engine_refresh(), virus_sigs_total(), virus_start(), and virus_stop().

4.50.2.115 `struct { ... } magma_t::web`

Referenced by api_response(), http_print_301(), http_response(), jansson_flags(), json_api_dispatch(), portal_endpoint(), portal_endpoint_error(), portal_endpoint_response(), portal_process(), and portal_upload().

4.50.2.116 `uint32_t magma_t::worker_threads`

Definition at line 54 of file global.h.

Referenced by queue_init(), queue_shutdown(), and queue_signal().

4.50.2.117 `uint32_t magma_t::wrap_line_length`

Definition at line 105 of file global.h.

Referenced by config_validate_settings(), and mail_message_cleanup().

4.51 mail_cache_t Struct Reference

```
#include <mail.h>
```

Data Fields

- uint64_t [messagenum](#)
- [stringer_t](#) * text

4.51.1 Detailed Description

Definition at line 14 of file mail.h.

4.51.2 Field Documentation

4.51.2.1 uint64_t mail_cache_t::messagenum

Definition at line 15 of file mail.h.

Referenced by [mail_cache_get\(\)](#), [mail_cache_reset\(\)](#), and [mail_cache_set\(\)](#).

4.51.2.2 stringer_t* mail_cache_t::text

Definition at line 16 of file mail.h.

Referenced by [mail_cache_destroy\(\)](#), [mail_cache_get\(\)](#), [mail_cache_reset\(\)](#), and [mail_cache_set\(\)](#).

4.52 mail_message_t Struct Reference

```
#include <mail.h>
```

Data Fields

- [placer_t](#) to
- [placer_t](#) date
- [placer_t](#) subject
- [stringer_t](#) * from
- [stringer_t](#) * text
- [mail_mime_t](#) * mime
- [size_t](#) [header_length](#)

4.52.1 Detailed Description

Definition at line 26 of file mail.h.

4.52.2 Field Documentation

4.52.2.1 [placer_t](#) mail_message_t::date

Definition at line 28 of file mail.h.

Referenced by [mail_message\(\)](#).

4.52.2.2 [stringer_t](#)* mail_message_t::from

Definition at line 30 of file mail.h.

Referenced by [mail_destroy\(\)](#), and [mail_message\(\)](#).

4.52.2.3 [size_t](#) mail_message_t::header_length

Definition at line 33 of file mail.h.

Referenced by [mail_load_header\(\)](#), [mail_message\(\)](#), and [portal_message_header\(\)](#).

4.52.2.4 [mail_mime_t](#)* mail_message_t::mime

Definition at line 32 of file mail.h.

Referenced by [imap_fetch_body\(\)](#), [imap_fetch_body_part\(\)](#), [imap_fetch_message\(\)](#), [mail_destroy\(\)](#), [mail_mime_update\(\)](#), [portal_message_attachments\(\)](#), and [portal_message_body\(\)](#).

4.52.2.5 [placer_t](#) mail_message_t::subject

Definition at line 29 of file mail.h.

Referenced by [mail_message\(\)](#).

4.52.2.6 stringer_t* mail_message_t::text

Definition at line 31 of file mail.h.

Referenced by imap_fetch_message(), mail_add_forward_headers(), mail_destroy(), mail_load_header(), mail_load_message(), mail_load_message_top(), mail_message(), mail_mime_update(), mail_modify_part(), mail_signature_add(), pop_retr(), pop_top(), and portal_message_header().

4.52.2.7 placer_t mail_message_t::to

Definition at line 27 of file mail.h.

Referenced by mail_message().

4.53 mail_mime_t Struct Reference

```
#include <mail.h>
```

Data Fields

- [array_t * children](#)
- [stringer_t * boundary](#)
- [uint32_t type](#)
- [uint32_t encoding](#)
- [placer_t header](#)
- [placer_t body](#)
- [placer_t entire](#)

4.53.1 Detailed Description

Definition at line 19 of file mail.h.

4.53.2 Field Documentation

4.53.2.1 `placer_t mail_mime_t::body`

Definition at line 23 of file mail.h.

Referenced by `imap_fetch_body()`, `imap_fetch_bodystructure()`, `imap_fetch_message()`, `mail_mime_part()`, `portal_message_attachments()`, and `portal_message_body()`.

4.53.2.2 `stringer_t* mail_mime_t::boundary`

Definition at line 21 of file mail.h.

Referenced by `mail_mime_free()`, and `mail_mime_part()`.

4.53.2.3 `array_t* mail_mime_t::children`

Definition at line 20 of file mail.h.

Referenced by `imap_fetch_body_part()`, `imap_fetch_bodystructure()`, `mail_mime_free()`, `mail_mime_part()`, `portal_message_attachments()`, and `portal_message_body()`.

4.53.2.4 `uint32_t mail_mime_t::encoding`

Definition at line 22 of file mail.h.

Referenced by `mail_mime_part()`.

4.53.2.5 `placer_t mail_mime_t::entire`

Definition at line 23 of file mail.h.

Referenced by `mail_mime_part()`.

4.53.2.6 placer_t mail_mime_t::header

Definition at line 23 of file mail.h.

Referenced by `imap_fetch_body()`, `imap_fetch_bodystructure()`, `mail_mime_part()`, and `portal_message_attachments()`.

4.53.2.7 uint32_t mail_mime_t::type

Definition at line 22 of file mail.h.

Referenced by `mail_mime_part()`, `portal_message_attachments()`, and `portal_message_body()`.

4.54 mappings_t Struct Reference

```
#include <encodings.h>
```

Data Fields

- struct {
 chr_t characters [32]
 chr_t values [128]
} zbase32
- struct {
 chr_t characters [64]
 chr_t values [128]
} base64
- struct {
 chr_t characters [64]
 chr_t values [128]
} base64_mod

4.54.1 Detailed Description

Definition at line 32 of file encodings.h.

4.54.2 Field Documentation

4.54.2.1 struct { ... } mappings_t::base64

Referenced by base64_decode(), base64_encode(), and base64_encode_wrap().

4.54.2.2 struct { ... } mappings_t::base64_mod

Referenced by base64_decode_mod(), and base64_encode_mod().

4.54.2.3 chr_t mappings_t::characters[64]

Definition at line 34 of file encodings.h.

Referenced by base64_encode(), base64_encode_mod(), base64_encode_wrap(), and zbase32_encode().

4.54.2.4 chr_t mappings_t::values[128] ()

Definition at line 34 of file encodings.h.

Referenced by base64_decode(), base64_decode_mod(), and zbase32_decode().

4.54.2.5 struct { ... } mappings_t::zbase32

Referenced by zbase32_decode(), and zbase32_encode().

4.55 media_type_t Struct Reference

```
#include <mail.h>
```

Data Fields

- [chr_t * extension](#)
- [bool_t bin](#)
- [chr_t * name](#)

4.55.1 Detailed Description

Definition at line 36 of file mail.h.

4.55.2 Field Documentation

4.55.2.1 bool_t media_type_t::bin

Definition at line 38 of file mail.h.

Referenced by mail_mime_encode_part().

4.55.2.2 chr_t* media_type_t::extension

Definition at line 37 of file mail.h.

4.55.2.3 chr_t* media_type_t::name

Definition at line 39 of file mail.h.

Referenced by mail_mime_encode_part().

4.56 meta_stats_tag_t Struct Reference

```
#include <meta.h>
```

Data Fields

- `uint64_t` [count](#)
- `stringer_t *` [tag](#)

4.56.1 Detailed Description

Definition at line 16 of file `meta.h`.

4.56.2 Field Documentation

4.56.2.1 `uint64_t` `meta_stats_tag_t::count`

Definition at line 17 of file `meta.h`.

Referenced by `meta_folders_stats_tags()`, and `portal_endpoint_folders_tags()`.

4.56.2.2 `stringer_t *` `meta_stats_tag_t::tag`

Definition at line 18 of file `meta.h`.

Referenced by `meta_folder_stats_tag_alloc()`, and `portal_endpoint_folders_tags()`.

4.57 multi_t Struct Reference

```
#include <strings.h>
```

Data Fields

- [M_TYPE](#) type
- union {
 - [bool_t](#) [binary](#)
 - void * [bl](#)
 - char * [ns](#)
 - [stringer_t](#) * [st](#)
 - uint8_t [u8](#)
 - uint16_t [u16](#)
 - uint32_t [u32](#)
 - uint64_t [u64](#)
 - int8_t [i8](#)
 - int16_t [i16](#)
 - int32_t [i32](#)
 - int64_t [i64](#)
 - float [fl](#)
 - double [dbl](#)
- [val](#)

4.57.1 Detailed Description

Definition at line 203 of file strings.h.

4.57.2 Field Documentation

4.57.2.1 bool_t multi_t::binary

Definition at line 206 of file strings.h.

Referenced by [cache_set_value\(\)](#), [cmp_mt_mt\(\)](#), [config_value_set\(\)](#), [ident_mt_mt\(\)](#), [mt_dupe\(\)](#), [mt_get_char\(\)](#), [relay_set_value\(\)](#), and [servers_set_value\(\)](#).

4.57.2.2 void* multi_t::bl

Definition at line 207 of file strings.h.

Referenced by [cache_output_help\(\)](#), [config_output_help\(\)](#), [mt_get_char\(\)](#), [mt_is_empty\(\)](#), [relay_output_help\(\)](#), and [servers_output_help\(\)](#).

4.57.2.3 double multi_t::dbl

Definition at line 219 of file strings.h.

Referenced by [cmp_mt_mt\(\)](#), [ident_mt_mt\(\)](#), [mt_dupe\(\)](#), [mt_get_char\(\)](#), and [mt_get_number\(\)](#).

4.57.2.4 float multi_t::fl

Definition at line 218 of file strings.h.

Referenced by `cmp_mt_mt()`, `ident_mt_mt()`, `mt_dupe()`, `mt_get_char()`, and `mt_get_number()`.

4.57.2.5 `int16_t multi_t::i16`

Definition at line 215 of file `strings.h`.

Referenced by `cache_validate()`, `cmp_mt_mt()`, `ident_mt_mt()`, `mt_dupe()`, `mt_get_char()`, `mt_get_number()`, `relay_validate()`, and `servers_validate()`.

4.57.2.6 `int32_t multi_t::i32`

Definition at line 216 of file `strings.h`.

Referenced by `cache_set_value()`, `cache_validate()`, `cmp_mt_mt()`, `config_value_set()`, `ident_mt_mt()`, `mt_dupe()`, `mt_get_char()`, `mt_get_number()`, `relay_set_value()`, `relay_validate()`, `servers_set_value()`, and `servers_validate()`.

4.57.2.7 `int64_t multi_t::i64`

Definition at line 217 of file `strings.h`.

Referenced by `cache_set_value()`, `cache_validate()`, `cmp_mt_mt()`, `config_value_set()`, `ident_mt_mt()`, `mt_dupe()`, `mt_get_char()`, `mt_get_number()`, `relay_set_value()`, `relay_validate()`, `servers_set_value()`, and `servers_validate()`.

4.57.2.8 `int8_t multi_t::i8`

Definition at line 214 of file `strings.h`.

Referenced by `cache_set_value()`, `cache_validate()`, `cmp_mt_mt()`, `config_value_set()`, `ident_mt_mt()`, `mt_dupe()`, `mt_get_char()`, `mt_get_number()`, `relay_set_value()`, `relay_validate()`, `servers_set_value()`, and `servers_validate()`.

4.57.2.9 `char* multi_t::ns`

Definition at line 208 of file `strings.h`.

Referenced by `cache_set_value()`, `cmp_mt_mt()`, `config_value_set()`, `ident_mt_mt()`, `mt_dupe()`, `mt_free()`, `mt_get_char()`, `mt_get_length()`, `mt_is_empty()`, `relay_set_value()`, and `servers_set_value()`.

4.57.2.10 `stringer_t* multi_t::st`

Definition at line 209 of file `strings.h`.

Referenced by `cache_set_value()`, `cmp_mt_mt()`, `config_load_cmdline_settings()`, `config_load_file_settings()`, `config_value_set()`, `contact_details_fetch()`, `contact_edit()`, `http_data_value_parse()`, `http_load_file()`, `http_parse_header()`, `ident_mt_mt()`, `meta_folders_stats_tags()`, `mt_dupe()`, `mt_free()`, `mt_get_char()`, `mt_get_length()`, `mt_is_empty()`, `nvp_parse()`, `relay_set_value()`, `servers_set_value()`, `teacher_add_cookie()`, `user_config_edit()`, `user_config_fetch()`, and `warehouse_fetch_domains()`.

4.57.2.11 `M_TYPE multi_t::type`

Definition at line 204 of file `strings.h`.

Referenced by `cache_free()`, `cache_output_settings()`, `cache_set_value()`, `cache_validate()`, `cmp_mt_mt()`, `config_free()`, `config_output_value()`, `config_value_set()`, `ident_mt_mt()`, `imap_message_copier()`, `meta_data_fetch_all_tags()`, `meta_data_fetch_folders()`, `meta_data_fetch_messages()`, `mt_dupe()`, `mt_free()`, `mt_get_char()`, `mt_get_length()`, `mt_get_null()`, `mt_get_number()`, `mt_get_type()`, `mt_is_empty()`, `mt_is_number()`, `mt_set_type()`, `relay_free()`, `relay_output_settings()`, `relay_set_value()`, `relay_validate()`, `servers_free()`, `servers_output_settings()`, `servers_set_value()`, `servers_validate()`, `signature_tree_add()`, and `smtp_add_bypass_entry()`.

4.57.2.12 uint16_t multi_t::u16

Definition at line 211 of file strings.h.

Referenced by cache_set_value(), cache_validate(), cmp_mt_mt(), config_value_set(), ident_mt_mt(), mt_dupe(), mt_get_char(), mt_get_number(), relay_set_value(), relay_validate(), servers_set_value(), and servers_validate().

4.57.2.13 uint32_t multi_t::u32

Definition at line 212 of file strings.h.

Referenced by cache_set_value(), cache_validate(), cmp_mt_mt(), config_value_set(), ident_mt_mt(), mt_dupe(), mt_get_char(), mt_get_number(), relay_set_value(), relay_validate(), servers_set_value(), and servers_validate().

4.57.2.14 uint64_t multi_t::u64

Definition at line 213 of file strings.h.

Referenced by cache_set_value(), cache_validate(), cmp_mt_mt(), config_value_set(), contact_folder_create(), contact_move(), contacts_fetch(), http_content_load_fonts(), ident_mt_mt(), imap_append_message(), imap_duplicate_messages(), imap_folder_create(), imap_folder_remove(), imap_folder_rename(), imap_message_copier(), imap_narrow_folders(), imap_narrow_messages(), imap_search_messages(), magma_folder_fetch(), message_folder_create(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), meta_messages_copier(), meta_messages_mover(), mt_dupe(), mt_get_char(), mt_get_number(), pop_session_destroy(), portal_endpoint_attachments_add(), portal_endpoint_attachments_remove(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_load(), portal_endpoint_contacts_remove(), portal_endpoint_messages_compose(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_load(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_get_upload_attachment(), portal_parse_json_str_array(), register_data_fetch_blocklist(), relay_set_value(), relay_validate(), servers_set_value(), servers_validate(), sess_create(), sess_get(), signature_tree_add(), smtp_add_bypass_entry(), smtp_fetch_inbound(), and warehouse_fetch_patterns().

4.57.2.15 uint8_t multi_t::u8

Definition at line 210 of file strings.h.

Referenced by cache_set_value(), cache_validate(), cmp_mt_mt(), config_value_set(), ident_mt_mt(), mt_dupe(), mt_get_char(), mt_get_number(), relay_set_value(), relay_validate(), servers_set_value(), and servers_validate().

4.57.2.16 union { ... } multi_t::val

Referenced by cache_output_help(), cache_set_value(), cache_validate(), cmp_mt_mt(), config_load_cmdline_settings(), config_load_file_settings(), config_output_help(), config_value_set(), contact_details_fetch(), contact_edit(), contact_folder_create(), contact_move(), contacts_fetch(), http_content_load_fonts(), http_data_value_parse(), http_load_file(), http_parse_header(), ident_mt_mt(), imap_append_message(), imap_duplicate_messages(), imap_folder_create(), imap_folder_remove(), imap_folder_rename(), imap_message_copier(), imap_narrow_folders(), imap_narrow_messages(), imap_search_messages(), magma_folder_fetch(), message_folder_create(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), meta_folders_stats_tags(), meta_messages_copier(), meta_messages_mover(), mt_dupe(), mt_free(), mt_get_char(), mt_get_length(), mt_get_number(), mt_is_empty(), nvp_parse(), pop_session_destroy(), portal_endpoint_attachments_add(), portal_endpoint_attachments_remove(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_load(), portal_endpoint_contacts_remove(), portal_endpoint_messages_compose(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_load(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_get_upload_attachment(), portal_parse_json_str_array(), register_data_fetch_blocklist(), relay_output_help(), relay_set_value(), relay_validate(), servers_output_help(), servers_set_value(), servers_validate(), sess_create(), sess_get(), signature_tree_add(), smtp_add_bypass_entry(), smtp_fetch_inbound(), teacher_add_cookie(), user_config_edit(), user_config_fetch(), warehouse_fetch_domains(), and warehouse_fetch_patterns().

4.58 mx_record_t Struct Reference

```
#include <dns.h>
```

Data Fields

- unsigned short [pref](#)
- char * [name](#)

4.58.1 Detailed Description

Definition at line 121 of file dns.h.

4.58.2 Field Documentation

4.58.2.1 char* mx_record_t::name

Definition at line 123 of file dns.h.

4.58.2.2 unsigned short mx_record_t::pref

Definition at line 122 of file dns.h.

4.59 nvp_t Struct Reference

```
#include <formats.h>
```

Public Member Functions

- [struct __attribute__ \(\(packed\)\)](#)

Data Fields

- [MAGMA_INDEX options](#)
- [tokens](#)
- [inx_t * pairs](#)

4.59.1 Detailed Description

Definition at line 13 of file formats.h.

4.59.2 Member Function Documentation

4.59.2.1 `struct nvp_t::__attribute__ ((packed))` [`inline`, `read`]

Definition at line 15 of file formats.h.

References comment.

4.59.3 Field Documentation

4.59.3.1 `MAGMA_INDEX nvp_t::options`

Definition at line 14 of file formats.h.

4.59.3.2 `inx_t* nvp_t::pairs`

Definition at line 18 of file formats.h.

Referenced by `config_load_cmdline_settings()`, `config_load_file_settings()`, `nvp_alloc()`, `nvp_free()`, and `nvp_parse()`.

4.59.3.3 `nvp_t::tokens`

Definition at line 17 of file formats.h.

Referenced by `nvp_alloc()`, and `nvp_parse()`.

4.60 object_cache_t Struct Reference

```
#include <objects.h>
```

Data Fields

- [inx_t](#) * meta
- [inx_t](#) * sessions

4.60.1 Detailed Description

Definition at line 30 of file objects.h.

4.60.2 Field Documentation

4.60.2.1 inx_t* object_cache_t::meta

Definition at line 31 of file objects.h.

Referenced by meta_inx_find(), meta_inx_remove(), obj_cache_prune(), obj_cache_start(), and obj_cache_stop().

4.60.2.2 inx_t * object_cache_t::sessions

Definition at line 31 of file objects.h.

Referenced by obj_cache_prune(), obj_cache_start(), obj_cache_stop(), sess_create(), and sess_get().

4.61 object_chunk Struct Reference

```
#include <crypto.h>
```

Data Fields

- struct [object_chunk](#) * [next](#)
- [dmime_chunk_type_t](#) [type](#)
- unsigned char [flags](#)
- size_t [data_size](#)
- unsigned char * [data](#)

4.61.1 Detailed Description

Definition at line 51 of file `crypto.h`.

4.61.2 Field Documentation

4.61.2.1 unsigned char* object_chunk::data

Definition at line 56 of file `crypto.h`.

4.61.2.2 size_t object_chunk::data_size

Definition at line 55 of file `crypto.h`.

4.61.2.3 unsigned char object_chunk::flags

Definition at line 54 of file `crypto.h`.

4.61.2.4 struct object_chunk* object_chunk::next [read]

Definition at line 52 of file `crypto.h`.

4.61.2.5 dmime_chunk_type_t object_chunk::type

Definition at line 53 of file `crypto.h`.

4.62 packedelem32_t Union Reference

```
#include <curve25519-donna-sse2.h>
```

Data Fields

- `uint32_t u` [4]
- `xmmi v`

4.62.1 Detailed Description

Definition at line 16 of file `curve25519-donna-sse2.h`.

4.62.2 Field Documentation

4.62.2.1 `uint32_t packedelem32_t::u[4]`

Definition at line 17 of file `curve25519-donna-sse2.h`.

4.62.2.2 `xmmi packedelem32_t::v`

Definition at line 18 of file `curve25519-donna-sse2.h`.

4.63 packedelem64_t Union Reference

```
#include <curve25519-donna-sse2.h>
```

Data Fields

- uint64_t [u](#) [2]
- [xmml](#) [v](#)

4.63.1 Detailed Description

Definition at line 21 of file curve25519-donna-sse2.h.

4.63.2 Field Documentation

4.63.2.1 uint64_t packedelem64_t::u[2]

Definition at line 22 of file curve25519-donna-sse2.h.

4.63.2.2 xmml packedelem64_t::v

Definition at line 23 of file curve25519-donna-sse2.h.

4.64 packedelem8_t Union Reference

```
#include <curve25519-donna-sse2.h>
```

Data Fields

- unsigned char [u](#) [16]
- [xmml](#) [v](#)

4.64.1 Detailed Description

Definition at line 11 of file curve25519-donna-sse2.h.

4.64.2 Field Documentation

4.64.2.1 unsigned char packedelem8_t::u[16]

Definition at line 12 of file curve25519-donna-sse2.h.

4.64.2.2 xmml packedelem8_t::v

Definition at line 13 of file curve25519-donna-sse2.h.

4.65 queue_t Struct Reference

Public Member Functions

- void(* [requeue](#) (void *[data](#)))(*)

Data Fields

- void(* [function](#))(void *[data](#))
- void(*)(*)* [data](#)
- struct [queue_t](#) * [next](#)

4.65.1 Detailed Description

Definition at line 10 of file queue.c.

4.65.2 Member Function Documentation

4.65.2.1 void(* queue_t::requeue (void * *data*)

Referenced by dequeue(), and requeue().

4.65.3 Field Documentation

4.65.3.1 void(*)(*) * queue_t::data

Definition at line 11 of file queue.c.

Referenced by dequeue(), and requeue().

4.65.3.2 void(* queue_t::function)(void **data*)

Referenced by dequeue(), and requeue().

4.65.3.3 struct queue_t* queue_t::next [\[read\]](#)

Definition at line 12 of file queue.c.

Referenced by dequeue(), and requeue().

4.66 random_data_t Struct Reference

Data Fields

- unsigned char [sk](#) [32]
- unsigned char [m](#) [128]

4.66.1 Detailed Description

Definition at line 116 of file fuzz-ed25519.c.

4.66.2 Field Documentation

4.66.2.1 unsigned char random_data_t::m[128]

Definition at line 118 of file fuzz-ed25519.c.

Referenced by `main()`.

4.66.2.2 unsigned char random_data_t::sk[32]

Definition at line 117 of file fuzz-ed25519.c.

Referenced by `main()`.

4.67 register_session_t Struct Reference

```
#include <register.h>
```

Data Fields

- uint64_t [usernum](#)
- uint16_t [plan](#)
- [stringer_t](#) * [username](#)
- [stringer_t](#) * [password](#)
- [stringer_t](#) * [hvf_value](#)
- [stringer_t](#) * [hvf_input](#)
- [stringer_t](#) * [name](#)

4.67.1 Detailed Description

Definition at line 17 of file register.h.

4.67.2 Field Documentation

4.67.2.1 [stringer_t](#) * [register_session_t::hvf_input](#)

Definition at line 20 of file register.h.

Referenced by [register_business_step1\(\)](#), [register_session_cache\(\)](#), [register_session_free\(\)](#), and [register_session_get\(\)](#).

4.67.2.2 [stringer_t](#) * [register_session_t::hvf_value](#)

Definition at line 20 of file register.h.

Referenced by [register_business_step1\(\)](#), [register_print_captcha\(\)](#), [register_session_cache\(\)](#), [register_session_free\(\)](#), and [register_session_get\(\)](#).

4.67.2.3 [stringer_t](#) * [register_session_t::name](#)

Definition at line 20 of file register.h.

Referenced by [register_print_step1\(\)](#), [register_print_step2\(\)](#), [register_session_cache\(\)](#), [register_session_free\(\)](#), [register_session_generate\(\)](#), and [register_session_get\(\)](#).

4.67.2.4 [stringer_t](#) * [register_session_t::password](#)

Definition at line 20 of file register.h.

Referenced by [register_business_step1\(\)](#), [register_business_step2\(\)](#), [register_session_cache\(\)](#), [register_session_free\(\)](#), and [register_session_get\(\)](#).

4.67.2.5 [uint16_t](#) [register_session_t::plan](#)

Definition at line 19 of file register.h.

Referenced by [register_business_step2\(\)](#), [register_session_cache\(\)](#), and [register_session_get\(\)](#).

4.67.2.6 stringer_t* register_session_t::username

Definition at line 20 of file register.h.

Referenced by register_business_step1(), register_business_step2(), register_print_step1(), register_session_cache(), register_session_free(), and register_session_get().

4.67.2.7 uint64_t register_session_t::usernum

Definition at line 18 of file register.h.

Referenced by register_business_step2(), register_process(), register_session_cache(), and register_session_get().

4.68 relay_keys_t Struct Reference

```
#include <relay.h>
```

Data Fields

- [size_t offset](#)
- [multi_t norm](#)
- [chr_t * name](#)
- [chr_t * description](#)
- [bool_t required](#)

4.68.1 Detailed Description

Definition at line 11 of file relay.h.

4.68.2 Field Documentation

4.68.2.1 chr_t* relay_keys_t::description

Definition at line 15 of file relay.h.

4.68.2.2 chr_t* relay_keys_t::name

Definition at line 14 of file relay.h.

Referenced by relay_set_value().

4.68.2.3 multi_t relay_keys_t::norm

Definition at line 13 of file relay.h.

Referenced by relay_free(), relay_output_help(), relay_output_settings(), relay_set_value(), and relay_validate().

4.68.2.4 size_t relay_keys_t::offset

Definition at line 12 of file relay.h.

Referenced by relay_free(), relay_output_settings(), relay_set_value(), and relay_validate().

4.68.2.5 bool_t relay_keys_t::required

Definition at line 16 of file relay.h.

Referenced by relay_output_help().

4.69 relay_t Struct Reference

```
#include <relay.h>
```

Data Fields

- [bool_t secure](#)
- [bool_t premium](#)
- [chr_t * name](#)
- [uint32_t port](#)

4.69.1 Detailed Description

Definition at line 19 of file relay.h.

4.69.2 Field Documentation

4.69.2.1 chr_t* relay_t::name

Definition at line 22 of file relay.h.

Referenced by smtp_client_connect().

4.69.2.2 uint32_t relay_t::port

Definition at line 23 of file relay.h.

Referenced by smtp_client_connect().

4.69.2.3 bool_t relay_t::premium

Definition at line 21 of file relay.h.

4.69.2.4 bool_t relay_t::secure

Definition at line 20 of file relay.h.

Referenced by smtp_client_connect().

4.70 server_config_t Struct Reference

```
#include <servers.h>
```

Data Fields

- SSL_CTX * [tls_ctx](#)
- char * [certificate](#)
- int [tls_sd](#)
- int [normal_sd](#)
- [stringer_t](#) * [name](#)
- [stringer_t](#) * [spam](#)
- [stringer_t](#) * [domain](#)
- [stringer_t](#) * [banner](#)
- unsigned [tls_port](#)
- unsigned [tcp_port](#)
- unsigned [listen_queue](#)
- unsigned [socket_timeout](#)
- unsigned [bad_command_cutoff](#)
- unsigned [bad_command_delay](#)
- struct [server_config_t](#) * [next](#)

4.70.1 Detailed Description

Definition at line 58 of file servers.h.

4.70.2 Field Documentation

4.70.2.1 unsigned server_config_t::bad_command_cutoff

Definition at line 63 of file servers.h.

4.70.2.2 unsigned server_config_t::bad_command_delay

Definition at line 63 of file servers.h.

4.70.2.3 stringer_t * server_config_t::banner

Definition at line 62 of file servers.h.

4.70.2.4 char* server_config_t::certificate

Definition at line 60 of file servers.h.

4.70.2.5 stringer_t * server_config_t::domain

Definition at line 62 of file servers.h.

4.70.2.6 unsigned server_config_t::listen_queue

Definition at line 63 of file servers.h.

4.70.2.7 stringer_t* server_config_t::name

Definition at line 62 of file servers.h.

4.70.2.8 struct server_config_t* server_config_t::next [read]

Definition at line 64 of file servers.h.

4.70.2.9 int server_config_t::normal_sd

Definition at line 61 of file servers.h.

4.70.2.10 unsigned server_config_t::socket_timeout

Definition at line 63 of file servers.h.

4.70.2.11 stringer_t * server_config_t::spam

Definition at line 62 of file servers.h.

4.70.2.12 unsigned server_config_t::tcp_port

Definition at line 63 of file servers.h.

4.70.2.13 SSL_CTX* server_config_t::tls_ctx

Definition at line 59 of file servers.h.

4.70.2.14 unsigned server_config_t::tls_port

Definition at line 63 of file servers.h.

4.70.2.15 int server_config_t::tls_sd

Definition at line 61 of file servers.h.

4.71 server_keys_t Struct Reference

```
#include <servers.h>
```

Data Fields

- [size_t offset](#)
- [multi_t norm](#)
- [chr_t * name](#)
- [chr_t * description](#)
- [bool_t required](#)

4.71.1 Detailed Description

Definition at line 27 of file servers.h.

4.71.2 Field Documentation

4.71.2.1 chr_t* server_keys_t::description

Definition at line 31 of file servers.h.

4.71.2.2 chr_t* server_keys_t::name

Definition at line 30 of file servers.h.

Referenced by servers_set_value().

4.71.2.3 multi_t server_keys_t::norm

Definition at line 29 of file servers.h.

Referenced by servers_free(), servers_output_help(), servers_output_settings(), servers_set_value(), and servers_validate().

4.71.2.4 size_t server_keys_t::offset

Definition at line 28 of file servers.h.

Referenced by servers_free(), servers_output_settings(), servers_set_value(), and servers_validate().

4.71.2.5 bool_t server_keys_t::required

Definition at line 32 of file servers.h.

Referenced by servers_output_help().

4.72 server_t Struct Reference

```
#include <servers.h>
```

Data Fields

- struct {
 SSL_CTX * context
 chr_t * certificate
} tls
- struct {
 int sockd
 bool_t ipv6
 uint32_t port
 uint32_t timeout
 uint32_t listen_queue
 M_PORT type
} network
- struct {
 uint32_t delay
 uint32_t cutoff
} violations
- bool_t enabled
- stringer_t * name
- stringer_t * domain
- M_PROTOCOL protocol

4.72.1 Detailed Description

Definition at line 35 of file servers.h.

4.72.2 Field Documentation

4.72.2.1 chr_t* server_t::certificate

Definition at line 38 of file servers.h.

Referenced by servers_encryption_start(), servers_validate(), and tls_server_create().

4.72.2.2 SSL_CTX* server_t::context

Definition at line 37 of file servers.h.

Referenced by protocol_process(), servers_encryption_stop(), tls_server_alloc(), tls_server_create(), and tls_server_destroy().

4.72.2.3 uint32_t server_t::cutoff

Definition at line 50 of file servers.h.

4.72.2.4 uint32_t server_t::delay

Definition at line 49 of file servers.h.

4.72.2.5 stringer_t * server_t::domain

Definition at line 53 of file servers.h.

Referenced by mail_build_signature(), and servers_network_start().

4.72.2.6 bool_t server_t::enabled

Definition at line 52 of file servers.h.

Referenced by net_listen(), net_trigger(), servers_encryption_start(), servers_encryption_stop(), servers_get_by_protocol(), servers_network_start(), and servers_network_stop().

4.72.2.7 bool_t server_t::ipv6

Definition at line 42 of file servers.h.

Referenced by net_init().

4.72.2.8 uint32_t server_t::listen_queue

Definition at line 45 of file servers.h.

Referenced by net_init().

4.72.2.9 stringer_t* server_t::name

Definition at line 53 of file servers.h.

Referenced by servers_network_start().

4.72.2.10 struct { ... } server_t::network

Referenced by con_init(), net_accept(), net_init(), net_listen(), net_shutdown(), net_trigger(), protocol_process(), servers_alloc(), servers_get_by_protocol(), servers_get_by_socket(), servers_get_count_using_port(), servers_network_stop(), and servers_validate().

4.72.2.11 uint32_t server_t::port

Definition at line 43 of file servers.h.

Referenced by net_init(), net_trigger(), servers_get_count_using_port(), and servers_validate().

4.72.2.12 M_PROTOCOL server_t::protocol

Definition at line 54 of file servers.h.

Referenced by servers_encryption_start(), servers_get_by_protocol(), and servers_validate().

4.72.2.13 int server_t::sockd

Definition at line 41 of file servers.h.

Referenced by con_init(), net_accept(), net_init(), net_listen(), net_shutdown(), net_trigger(), servers_alloc(), servers_get_by_socket(), and servers_network_stop().

4.72.2.14 uint32_t server_t::timeout

Definition at line 44 of file servers.h.

Referenced by protocol_process().

4.72.2.15 struct { ... } server_t::tls

Referenced by protocol_process(), servers_encryption_start(), servers_encryption_stop(), servers_validate(), tls_server_alloc(), tls_server_create(), and tls_server_destroy().

4.72.2.16 M_PORT server_t::type

Definition at line 46 of file servers.h.

Referenced by protocol_process(), servers_get_by_protocol(), and servers_validate().

4.72.2.17 struct { ... } server_t::violations

4.73 sha_databuf_t Struct Reference

```
#include <misc.h>
```

Data Fields

- void * [data](#)
- size_t [len](#)

4.73.1 Detailed Description

Definition at line 36 of file misc.h.

4.73.2 Field Documentation

4.73.2.1 void* sha_databuf_t::data

Definition at line 37 of file misc.h.

4.73.2.2 size_t sha_databuf_t::len

Definition at line 38 of file misc.h.

4.74 `signet_field_key_t` Struct Reference

```
#include <common.h>
```

Data Fields

- unsigned int `required`
- unsigned int `unique`
- unsigned char `bytes_name_size`
- unsigned char `bytes_data_size`
- uint32_t `data_size`
- `field_data_t` `data_type`
- const char * `name`
- const char * `description`

4.74.1 Detailed Description

Definition at line 150 of file `common.h`.

4.74.2 Field Documentation

4.74.2.1 unsigned char `signet_field_key_t::bytes_data_size`

Number of bytes for this

Definition at line 156 of file `common.h`.

4.74.2.2 unsigned char `signet_field_key_t::bytes_name_size`

Is this a defined field

Definition at line 155 of file `common.h`.

4.74.2.3 uint32_t `signet_field_key_t::data_size`

`data_size = 0` indicates the size being variable

Definition at line 157 of file `common.h`.

4.74.2.4 `field_data_t` `signet_field_key_t::data_type`

Dump format for the field

Definition at line 159 of file `common.h`.

4.74.2.5 const char* `signet_field_key_t::description`

field type description

Definition at line 162 of file `common.h`.

4.74.2.6 `const char* signet_field_key_t::name`

Definition at line 161 of file `common.h`.

4.74.2.7 `unsigned int signet_field_key_t::required`

is this field required

Definition at line 152 of file `common.h`.

4.74.2.8 `unsigned int signet_field_key_t::unique`

can there be multiple fields of this identifier

Definition at line 153 of file `common.h`.

4.75 `signet_field_t` Struct Reference

Data Fields

- const `signet_t * signet`
- `signet_field_key_t * key`
- unsigned char `name_size`
- unsigned int `data_size`
- unsigned int `id_offset`
- unsigned int `name_offset`
- unsigned int `data_offset`
- struct `signet_field_t * next`

4.75.1 Detailed Description

A signet field index structure for temporary convenience organization of field data

Definition at line 7 of file `signet.c`.

4.75.2 Field Documentation

4.75.2.1 unsigned int `signet_field_t::data_offset`

Definition at line 16 of file `signet.c`.

4.75.2.2 unsigned int `signet_field_t::data_size`

Definition at line 12 of file `signet.c`.

4.75.2.3 unsigned int `signet_field_t::id_offset`

Definition at line 14 of file `signet.c`.

4.75.2.4 `signet_field_key_t*` `signet_field_t::key`

Definition at line 10 of file `signet.c`.

4.75.2.5 unsigned int `signet_field_t::name_offset`

Definition at line 15 of file `signet.c`.

4.75.2.6 unsigned char `signet_field_t::name_size`

Definition at line 11 of file `signet.c`.

4.75.2.7 struct `signet_field_t*` `signet_field_t::next` `[read]`

Definition at line 18 of file `signet.c`.

4.75.2.8 const signet_t* signet_field_t::signet

Definition at line 9 of file signet.c.

4.76 `signet_t` Struct Reference

```
#include <signet.h>
```

Data Fields

- [signet_type_t](#) type
- [uint32_t](#) [fields](#) [256]
- [uint32_t](#) [size](#)
- unsigned char * [data](#)

4.76.1 Detailed Description

Definition at line 27 of file `signet.h`.

4.76.2 Field Documentation

4.76.2.1 `unsigned char* signet_t::data`

Definition at line 32 of file `signet.h`.

4.76.2.2 `uint32_t signet_t::fields[256]`

Each index corresponds to a different field type identifier. The value of `fields[index]` is the byte directly after the first occurrence of the corresponding field type identifier. If `fields[index]` is 0 it means that the corresponding field type identifier occurred 0 times.

Definition at line 29 of file `signet.h`.

4.76.2.3 `uint32_t signet_t::size`

Combined length of all the fields

Definition at line 31 of file `signet.h`.

4.76.2.4 `signet_type_t signet_t::type`

Definition at line 28 of file `signet.h`.

4.77 smtp_inbound_filter_t Struct Reference

```
#include <smtp.h>
```

Data Fields

- uint64_t [foldernum](#)
- uint64_t [rulenum](#)
- unsigned [location](#)
- unsigned [type](#)
- unsigned [action](#)
- [stringer_t](#) * [field](#)
- [stringer_t](#) * [label](#)
- [stringer_t](#) * [expression](#)

4.77.1 Detailed Description

Definition at line 75 of file smtp.h.

4.77.2 Field Documentation

4.77.2.1 unsigned smtp_inbound_filter_t::action

Definition at line 77 of file smtp.h.

Referenced by [smtp_fetch_inbound\(\)](#).

4.77.2.2 stringer_t * smtp_inbound_filter_t::expression

Definition at line 78 of file smtp.h.

Referenced by [smtp_fetch_inbound\(\)](#), and [smtp_list_free_filter\(\)](#).

4.77.2.3 stringer_t * smtp_inbound_filter_t::field

Definition at line 78 of file smtp.h.

Referenced by [smtp_fetch_inbound\(\)](#), and [smtp_list_free_filter\(\)](#).

4.77.2.4 uint64_t smtp_inbound_filter_t::foldernum

Definition at line 76 of file smtp.h.

Referenced by [smtp_fetch_inbound\(\)](#).

4.77.2.5 stringer_t * smtp_inbound_filter_t::label

Definition at line 78 of file smtp.h.

Referenced by [smtp_fetch_inbound\(\)](#), and [smtp_list_free_filter\(\)](#).

4.77.2.6 unsigned smtp_inbound_filter_t::location

Definition at line 77 of file smtp.h.

Referenced by smtp_fetch_inbound().

4.77.2.7 uint64_t smtp_inbound_filter_t::rulenum

Definition at line 76 of file smtp.h.

Referenced by smtp_fetch_inbound().

4.77.2.8 unsigned smtp_inbound_filter_t::type

Definition at line 77 of file smtp.h.

Referenced by smtp_fetch_inbound().

4.78 smtp_inbound_prefs_t Struct Reference

```
#include <smtp.h>
```

Data Fields

- [int_t](#) outcome
- [prime_t](#) * signet
- [size_t](#) rcv_size_limit
- [stringer_t](#) * rcptto
- [stringer_t](#) * address
- [stringer_t](#) * domain
- [stringer_t](#) * forwarded
- [stringer_t](#) * spamsig
- [uint32_t](#) greytime
- [uint32_t](#) local_size
- [uint32_t](#) daily_rcv_limit
- [uint32_t](#) daily_rcv_limit_ip
- [uint64_t](#) usernum
- [uint64_t](#) signum
- [uint64_t](#) spamkey
- [uint64_t](#) quota
- [uint64_t](#) stor_size
- [uint64_t](#) inbox
- [uint64_t](#) autoreply
- [uint64_t](#) messagenum
- [uint64_t](#) foldernum
- [int_t](#) mark
- [int_t](#) secure
- [int_t](#) rollout
- [int_t](#) spam
- [int_t](#) virus
- [int_t](#) greylist
- [int_t](#) spf
- [int_t](#) dkim
- [int_t](#) rbl
- [int_t](#) phish
- [int_t](#) overquota
- [int_t](#) bounces
- [int_t](#) spfaction
- [int_t](#) dkimaction
- [int_t](#) rblaction
- [int_t](#) spamaction
- [int_t](#) virusaction
- [int_t](#) phishaction
- [int_t](#) spam_checked
- [inx_t](#) * filters
- [struct](#) smtp_inbound_prefs_t * next

4.78.1 Detailed Description

Definition at line 82 of file smtp.h.

4.78.2 Field Documentation

4.78.2.1 `stringer_t * smtp_inbound_prefs_t::address`

Definition at line 86 of file smtp.h.

Referenced by `smtp_fetch_inbound()`, and `smtp_free_inbound()`.

4.78.2.2 `uint64_t smtp_inbound_prefs_t::autoreply`

Definition at line 88 of file smtp.h.

Referenced by `smtp_accept_message()`, and `smtp_fetch_inbound()`.

4.78.2.3 `int_t smtp_inbound_prefs_t::bounces`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_fetch_inbound()`, and `smtp_update_receive_stats()`.

4.78.2.4 `uint32_t smtp_inbound_prefs_t::daily_recv_limit`

Definition at line 87 of file smtp.h.

Referenced by `smtp_check_receive_quota()`, `smtp_fetch_inbound()`, and `smtp_rcpt_to()`.

4.78.2.5 `uint32_t smtp_inbound_prefs_t::daily_recv_limit_ip`

Definition at line 87 of file smtp.h.

Referenced by `smtp_check_receive_quota()`, `smtp_fetch_inbound()`, and `smtp_rcpt_to()`.

4.78.2.6 `int_t smtp_inbound_prefs_t::dkim`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, and `smtp_fetch_inbound()`.

4.78.2.7 `int_t smtp_inbound_prefs_t::dkimaction`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, and `smtp_fetch_inbound()`.

4.78.2.8 `stringer_t * smtp_inbound_prefs_t::domain`

Definition at line 86 of file smtp.h.

Referenced by `smtp_fetch_inbound()`, and `smtp_free_inbound()`.

4.78.2.9 `inx_t* smtp_inbound_prefs_t::filters`

Definition at line 91 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_check_filters()`, `smtp_fetch_inbound()`, and `smtp_free_inbound()`.

4.78.2.10 uint64_t smtp_inbound_prefs_t::foldernum

Definition at line 88 of file smtp.h.

Referenced by smtp_accept_message(), smtp_check_filters(), and smtp_store_message().

4.78.2.11 stringer_t * smtp_inbound_prefs_t::forwarded

Definition at line 86 of file smtp.h.

Referenced by smtp_accept_message(), smtp_fetch_inbound(), smtp_free_inbound(), and smtp_rcpt_to().

4.78.2.12 int_t smtp_inbound_prefs_t::greylist

Definition at line 89 of file smtp.h.

Referenced by smtp_fetch_inbound(), and smtp_rcpt_to().

4.78.2.13 uint32_t smtp_inbound_prefs_t::greytime

Definition at line 87 of file smtp.h.

Referenced by smtp_check_greylist(), smtp_fetch_inbound(), and smtp_rcpt_to().

4.78.2.14 uint64_t smtp_inbound_prefs_t::inbox

Definition at line 88 of file smtp.h.

Referenced by smtp_accept_message(), and smtp_fetch_inbound().

4.78.2.15 uint32_t smtp_inbound_prefs_t::local_size

Definition at line 87 of file smtp.h.

4.78.2.16 int_t smtp_inbound_prefs_t::mark

Definition at line 89 of file smtp.h.

Referenced by smtp_accept_message(), smtp_check_filters(), and smtp_store_message().

4.78.2.17 uint64_t smtp_inbound_prefs_t::messagenum

Definition at line 88 of file smtp.h.

Referenced by smtp_store_message().

4.78.2.18 struct smtp_inbound_prefs_t * smtp_inbound_prefs_t::next [read]

Definition at line 92 of file smtp.h.

Referenced by mail_add_required_headers(), smtp_add_inbound(), smtp_bounce(), smtp_check_duplicate_recipient(), smtp_data_inbound(), and smtp_free_inbound().

4.78.2.19 `int_t smtp_inbound_prefs_t::outcome`

Definition at line 83 of file smtp.h.

Referenced by `smtp_bounce()`, and `smtp_data_inbound()`.

4.78.2.20 `int_t smtp_inbound_prefs_t::overquota`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_fetch_inbound()`, and `smtp_rcpt_to()`.

4.78.2.21 `int_t smtp_inbound_prefs_t::phish`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, and `smtp_fetch_inbound()`.

4.78.2.22 `int_t smtp_inbound_prefs_t::phishaction`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, and `smtp_fetch_inbound()`.

4.78.2.23 `uint64_t smtp_inbound_prefs_t::quota`

Definition at line 88 of file smtp.h.

Referenced by `smtp_fetch_inbound()`, and `smtp_rollout()`.

4.78.2.24 `int_t smtp_inbound_prefs_t::rbl`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_fetch_inbound()`, and `smtp_rcpt_to()`.

4.78.2.25 `int_t smtp_inbound_prefs_t::rblaction`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_fetch_inbound()`, and `smtp_rcpt_to()`.

4.78.2.26 `stringer_t* smtp_inbound_prefs_t::rcptto`

Definition at line 86 of file smtp.h.

Referenced by `mail_add_inbound_headers()`, `mail_add_required_headers()`, `smtp_accept_message()`, `smtp_bounce()`, `smtp_fetch_inbound()`, `smtp_free_inbound()`, and `smtp_rcpt_to()`.

4.78.2.27 `size_t smtp_inbound_prefs_t::recv_size_limit`

Definition at line 85 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_fetch_inbound()`, and `smtp_rcpt_to()`.

4.78.2.28 int_t smtp_inbound_prefs_t::rollout

Definition at line 89 of file smtp.h.

Referenced by smtp_accept_message(), smtp_fetch_inbound(), and smtp_rcpt_to().

4.78.2.29 int_t smtp_inbound_prefs_t::secure

Definition at line 89 of file smtp.h.

Referenced by smtp_fetch_inbound().

4.78.2.30 prime_t* smtp_inbound_prefs_t::signet

Definition at line 84 of file smtp.h.

Referenced by smtp_fetch_inbound(), smtp_free_inbound(), and smtp_store_message().

4.78.2.31 uint64_t smtp_inbound_prefs_t::signum

Definition at line 88 of file smtp.h.

Referenced by smtp_accept_message(), smtp_store_message(), and smtp_store_spamsig().

4.78.2.32 int_t smtp_inbound_prefs_t::spam

Definition at line 89 of file smtp.h.

Referenced by smtp_accept_message(), and smtp_fetch_inbound().

4.78.2.33 int_t smtp_inbound_prefs_t::spam_checked

Definition at line 89 of file smtp.h.

Referenced by smtp_accept_message().

4.78.2.34 int_t smtp_inbound_prefs_t::spamaction

Definition at line 89 of file smtp.h.

Referenced by smtp_accept_message(), and smtp_fetch_inbound().

4.78.2.35 uint64_t smtp_inbound_prefs_t::spamkey

Definition at line 88 of file smtp.h.

Referenced by smtp_accept_message(), smtp_store_message(), and smtp_store_spamsig().

4.78.2.36 stringer_t * smtp_inbound_prefs_t::spamsig

Definition at line 86 of file smtp.h.

Referenced by smtp_accept_message(), smtp_free_inbound(), and smtp_insert_spamsig().

4.78.2.37 `int_t smtp_inbound_prefs_t::spf`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_fetch_inbound()`, and `smtp_rcpt_to()`.

4.78.2.38 `int_t smtp_inbound_prefs_t::spfaction`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_fetch_inbound()`, and `smtp_rcpt_to()`.

4.78.2.39 `uint64_t smtp_inbound_prefs_t::stor_size`

Definition at line 88 of file smtp.h.

Referenced by `smtp_fetch_inbound()`, and `smtp_rollout()`.

4.78.2.40 `uint64_t smtp_inbound_prefs_t::usernum`

Definition at line 88 of file smtp.h.

Referenced by `smtp_accept_message()`, `smtp_check_duplicate_recipient()`, `smtp_check_filters()`, `smtp_check_greylist()`, `smtp_check_receive_quota()`, `smtp_fetch_inbound()`, `smtp_insert_spamsig()`, `smtp_rcpt_to()`, `smtp_rollout()`, `smtp_store_message()`, and `smtp_update_receive_stats()`.

4.78.2.41 `int_t smtp_inbound_prefs_t::virus`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, and `smtp_fetch_inbound()`.

4.78.2.42 `int_t smtp_inbound_prefs_t::virusaction`

Definition at line 89 of file smtp.h.

Referenced by `smtp_accept_message()`, and `smtp_fetch_inbound()`.

4.79 smtp_message_t Struct Reference

```
#include <smtp.h>
```

Data Fields

- [placer_t](#) to
- [placer_t](#) date
- [placer_t](#) subject
- [stringer_t](#) * id
- [stringer_t](#) * from
- [stringer_t](#) * text
- [size_t](#) header_length

4.79.1 Detailed Description

Definition at line 59 of file smtp.h.

4.79.2 Field Documentation

4.79.2.1 [placer_t](#) smtp_message_t::date

Definition at line 61 of file smtp.h.

Referenced by mail_add_required_headers(), and mail_headers().

4.79.2.2 [stringer_t](#)* smtp_message_t::from

Definition at line 64 of file smtp.h.

Referenced by mail_add_required_headers(), mail_destroy_message(), and mail_headers().

4.79.2.3 [size_t](#) smtp_message_t::header_length

Definition at line 66 of file smtp.h.

Referenced by mail_add_required_headers(), mail_create_message(), and mail_headers().

4.79.2.4 [stringer_t](#)* smtp_message_t::id

Definition at line 63 of file smtp.h.

Referenced by mail_create_message(), and mail_destroy_message().

4.79.2.5 [placer_t](#) smtp_message_t::subject

Definition at line 62 of file smtp.h.

Referenced by mail_add_required_headers(), and mail_headers().

4.79.2.6 stringer_t* smtp_message_t::text

Definition at line 65 of file smtp.h.

Referenced by mail_add_required_headers(), mail_create_message(), mail_destroy_message(), and mail_headers().

4.79.2.7 placer_t smtp_message_t::to

Definition at line 60 of file smtp.h.

Referenced by mail_add_required_headers(), and mail_headers().

4.80 smtp_outbound_prefs_t Struct Reference

```
#include <smtp.h>
```

Data Fields

- uint64_t [usernum](#)
- [stringer_t](#) * [domain](#)
- [int_t](#) [tls](#)
- [int_t](#) [importance](#)
- uint32_t [sent_today](#)
- uint32_t [daily_send_limit](#)
- uint32_t [send_size_limit](#)
- [smtp_recipients_t](#) * [recipients](#)

4.80.1 Detailed Description

Definition at line 96 of file smtp.h.

4.80.2 Field Documentation

4.80.2.1 uint32_t smtp_outbound_prefs_t::daily_send_limit

Definition at line 100 of file smtp.h.

Referenced by [smtp_check_transmit_quota\(\)](#), and [smtp_fetch_authorization\(\)](#).

4.80.2.2 stringer_t* smtp_outbound_prefs_t::domain

Definition at line 98 of file smtp.h.

Referenced by [smtp_fetch_authorization\(\)](#), and [smtp_free_outbound\(\)](#).

4.80.2.3 int_t smtp_outbound_prefs_t::importance

Definition at line 99 of file smtp.h.

Referenced by [smtp_fetch_authorization\(\)](#).

4.80.2.4 smtp_recipients_t* smtp_outbound_prefs_t::recipients

Definition at line 101 of file smtp.h.

Referenced by [smtp_free_outbound\(\)](#).

4.80.2.5 uint32_t smtp_outbound_prefs_t::send_size_limit

Definition at line 100 of file smtp.h.

Referenced by [smtp_auth_login\(\)](#), [smtp_auth_plain\(\)](#), and [smtp_fetch_authorization\(\)](#).

4.80.2.6 uint32_t smtp_outbound_prefs_t::sent_today

Definition at line 100 of file smtp.h.

Referenced by smtp_check_transmit_quota(), and smtp_fetch_authorization().

4.80.2.7 int_t smtp_outbound_prefs_t::tls

Definition at line 99 of file smtp.h.

Referenced by smtp_fetch_authorization().

4.80.2.8 uint64_t smtp_outbound_prefs_t::usernum

Definition at line 97 of file smtp.h.

Referenced by smtp_fetch_authorization().

4.81 smtp_recipients_t Struct Reference

```
#include <smtp.h>
```

Data Fields

- [stringer_t](#) * address
- struct [smtp_recipients_t](#) * next

4.81.1 Detailed Description

Definition at line 70 of file smtp.h.

4.81.2 Field Documentation

4.81.2.1 stringer_t* smtp_recipients_t::address

Definition at line 71 of file smtp.h.

Referenced by mail_add_required_headers(), smtp_add_recipient(), smtp_free_recipients(), and smtp_relay_message().

4.81.2.2 struct smtp_recipients_t* smtp_recipients_t::next [read]

Definition at line 72 of file smtp.h.

Referenced by mail_add_required_headers(), smtp_add_recipient(), smtp_free_recipients(), and smtp_relay_message().

4.82 smtp_session_t Struct Reference

```
#include <smtp.h>
```

Data Fields

- [bool_t esmtp](#)
- [bool_t submission](#)
- [bool_t authenticated](#)
- [bool_t suggested_eight_bit](#)
- [size_t max_length](#)
- [size_t num_recipients](#)
- [size_t suggested_length](#)
- [stringer_t * helo](#)
- [stringer_t * mailfrom](#)
- [uint32_t ip_address_v4](#)
- [struct {](#)
 - [int_t rbl](#)
 - [int_t spf](#)
 - [int_t dkim](#)
 - [int_t virus](#)[} checked](#)
- [smtp_message_t * message](#)
- [smtp_inbound_prefs_t * in_prefs](#)
- [smtp_outbound_prefs_t * out_prefs](#)
- [bool_t bypass](#)

4.82.1 Detailed Description

Definition at line 104 of file smtp.h.

4.82.2 Field Documentation

4.82.2.1 bool_t smtp_session_t::authenticated

Definition at line 107 of file smtp.h.

4.82.2.2 bool_t smtp_session_t::bypass

Definition at line 130 of file smtp.h.

4.82.2.3 struct { ... } smtp_session_t::checked

4.82.2.4 int_t smtp_session_t::dkim

Definition at line 122 of file smtp.h.

4.82.2.5 bool_t smtp_session_t::esmtp

Definition at line 105 of file smtp.h.

4.82.2.6 stringer_t* smtp_session_t::helo

Definition at line 114 of file smtp.h.

4.82.2.7 smtp_inbound_prefs_t* smtp_session_t::in_prefs

Definition at line 127 of file smtp.h.

4.82.2.8 uint32_t smtp_session_t::ip_address_v4

Definition at line 117 of file smtp.h.

4.82.2.9 stringer_t* smtp_session_t::mailfrom

Definition at line 115 of file smtp.h.

4.82.2.10 size_t smtp_session_t::max_length

Definition at line 110 of file smtp.h.

4.82.2.11 smtp_message_t* smtp_session_t::message

Definition at line 126 of file smtp.h.

4.82.2.12 size_t smtp_session_t::num_recipients

Definition at line 111 of file smtp.h.

4.82.2.13 smtp_outbound_prefs_t* smtp_session_t::out_prefs

Definition at line 128 of file smtp.h.

4.82.2.14 int_t smtp_session_t::rbl

Definition at line 120 of file smtp.h.

4.82.2.15 int_t smtp_session_t::spf

Definition at line 121 of file smtp.h.

4.82.2.16 bool_t smtp_session_t::submission

Definition at line 106 of file smtp.h.

4.82.2.17 bool_t smtp_session_t::suggested_eight_bit

Definition at line 108 of file smtp.h.

4.82.2.18 `size_t smtp_session_t::suggested_length`

Definition at line 112 of file smtp.h.

4.82.2.19 `int_t smtp_session_t::virus`

Definition at line 123 of file smtp.h.

4.83 statistics_vp_t Struct Reference

```
#include <statistics.h>
```

Data Fields

- MYSQL_STMT ** [stmt](#)
- uint64_t [val](#)

4.83.1 Detailed Description

Definition at line 26 of file statistics.h.

4.83.2 Field Documentation

4.83.2.1 MYSQL_STMT** statistics_vp_t::stmt

Definition at line 27 of file statistics.h.

Referenced by [statistics_init\(\)](#), and [statistics_refresh\(\)](#).

4.83.2.2 uint64_t statistics_vp_t::val

Definition at line 28 of file statistics.h.

Referenced by [statistics_process\(\)](#), and [statistics_refresh\(\)](#).

4.84 subnet_t Struct Reference

```
#include <host.h>
```

Data Fields

- [uint32_t mask](#)
- [ip_t address](#)

4.84.1 Detailed Description

Definition at line 80 of file host.h.

4.84.2 Field Documentation

4.84.2.1 ip_t subnet_t::address

Definition at line 82 of file host.h.

Referenced by `ip_matches_subnet()`, and `ip_subnet_st()`.

4.84.2.2 uint32_t subnet_t::mask

Definition at line 81 of file host.h.

Referenced by `ip_matches_subnet()`, and `ip_subnet_st()`.

4.85 teacher_data_t Struct Reference

```
#include <teacher.h>
```

Data Fields

- [int_t](#) completed
- [int_t](#) disposition
- [uint64_t](#) usernum
- [uint64_t](#) signum
- [uint64_t](#) keynum
- [stringer_t](#) * signature
- [stringer_t](#) * username
- [stringer_t](#) * password

4.85.1 Detailed Description

Definition at line 12 of file teacher.h.

4.85.2 Field Documentation

4.85.2.1 int_t teacher_data_t::completed

Definition at line 13 of file teacher.h.

Referenced by teacher_data_delete(), teacher_data_get(), teacher_data_save(), and teacher_process().

4.85.2.2 int_t teacher_data_t::disposition

Definition at line 13 of file teacher.h.

Referenced by teacher_data_delete(), teacher_data_fetch(), teacher_data_get(), teacher_data_save(), teacher_print_form(), and teacher_process().

4.85.2.3 uint64_t teacher_data_t::keynum

Definition at line 14 of file teacher.h.

Referenced by teacher_data_fetch(), teacher_data_get(), teacher_data_save(), teacher_print_form(), and teacher_process().

4.85.2.4 stringer_t * teacher_data_t::password

Definition at line 15 of file teacher.h.

Referenced by teacher_add_cookie(), teacher_data_delete(), teacher_data_fetch(), teacher_data_free(), teacher_data_get(), teacher_data_save(), and teacher_process().

4.85.2.5 stringer_t * teacher_data_t::signature

Definition at line 15 of file teacher.h.

Referenced by teacher_data_delete(), teacher_data_fetch(), teacher_data_free(), teacher_data_get(), teacher_data_save(), and teacher_process().

4.85.2.6 uint64_t teacher_data_t::signum

Definition at line 14 of file teacher.h.

Referenced by teacher_data_delete(), teacher_data_fetch(), teacher_data_get(), teacher_data_save(), and teacher_print_form().

4.85.2.7 stringer_t * teacher_data_t::username

Definition at line 15 of file teacher.h.

Referenced by teacher_data_delete(), teacher_data_fetch(), teacher_data_free(), teacher_data_get(), teacher_data_save(), and teacher_process().

4.85.2.8 uint64_t teacher_data_t::usernum

Definition at line 14 of file teacher.h.

Referenced by teacher_data_delete(), teacher_data_fetch(), teacher_data_get(), teacher_data_save(), and teacher_process().

4.86 test_data_t Struct Reference

Data Fields

- unsigned char [sk](#) [32]
- unsigned char [pk](#) [32]
- unsigned char [sig](#) [64]
- const char * [m](#)

4.86.1 Detailed Description

Definition at line 50 of file test.c.

4.86.2 Field Documentation

4.86.2.1 const char* test_data_t::m

Definition at line 52 of file test.c.

4.86.2.2 unsigned char test_data_t::pk[32]

Definition at line 51 of file test.c.

4.86.2.3 unsigned char test_data_t::sig[64]

Definition at line 51 of file test.c.

4.86.2.4 unsigned char test_data_t::sk[32]

Definition at line 51 of file test.c.

4.87 tok_state_t Struct Reference

```
#include <parsers.h>
```

Data Fields

- char [token](#)
- char * [block](#)
- char * [position](#)
- size_t [length](#)
- size_t [remaining](#)

4.87.1 Detailed Description

Definition at line 11 of file parsers.h.

4.87.2 Field Documentation

4.87.2.1 char* tok_state_t::block

Definition at line 13 of file parsers.h.

Referenced by tok_pop_init_bl(), and tok_pop_init_st().

4.87.2.2 size_t tok_state_t::length

Definition at line 14 of file parsers.h.

Referenced by tok_pop_init_bl(), and tok_pop_init_st().

4.87.2.3 char * tok_state_t::position

Definition at line 13 of file parsers.h.

Referenced by tok_pop(), tok_pop_init_bl(), and tok_pop_init_st().

4.87.2.4 size_t tok_state_t::remaining

Definition at line 14 of file parsers.h.

Referenced by tok_pop(), tok_pop_init_bl(), and tok_pop_init_st().

4.87.2.5 char tok_state_t::token

Definition at line 12 of file parsers.h.

Referenced by tok_pop(), tok_pop_init_bl(), and tok_pop_init_st().

Chapter 5

File Documentation

5.1 src/core/buckets/arrays.c File Reference

```
#include "magma.h"
```

Functions

- `array_t * ar_alloc (size_t size)`
Allocate an array of a specified number of elements.
- `size_t ar_avail_get (array_t *array)`
Get the maximum number of elements allocated in an array.
- `size_t ar_length_get (array_t *array)`
Get the number of elements actively stored in an array.
- `uint32_t ar_field_type (array_t *array, size_t element)`
- `stringer_t * ar_field_st (array_t *array, size_t element)`
Return the value of an element in an array that holds a managed string.
- `chr_t * ar_field_ns (array_t *array, size_t element)`
Return the value of an element in an array that holds a null-terminated string.
- `array_t * ar_field_ar (array_t *array, size_t element)`
Return the value of an element in an array that holds another array.
- `placer_t * ar_field_pl (array_t *array, size_t element)`
Return the value of an element in an array that holds a placer.
- `void * ar_field_ptr (array_t *array, size_t element)`
Return the value of an element in an array that holds a pointer.
- `void ar_length_set (array_t *array, size_t used)`
Manually set the number of used elements in an array.
- `int_t ar_append (array_t **array, uint32_t type, void *item)`
- `void ar_free (array_t *array)`
Free an array object and all of its underlying elements.

- `array_t * ar_dupe (array_t *array)`

Create a duplicate copy of the array.

5.1.1 Function Documentation

5.1.1.1 `array_t* ar_alloc (size_t size)`

Allocate an array of a specified number of elements. [arrays.c](#)

Parameters:

size the number of elements the array is capable of holding.

Returns:

NULL on failure, or a pointer to the newly allocated array on success.

Definition at line 15 of file `arrays.c`.

References `ARRAY_MAX_ELEMENTS`, `log_error`, `log_pedantic`, and `mm_wipe()`.

Referenced by `ar_append()`, `ar_dupe()`, `mail_mime_part()`, `mail_mime_split()`, `mail_mime_type_parameters()`, and `meta_data_fetch_message_tags()`.

5.1.1.2 `int_t ar_append (array_t **array, uint32_t type, void *item)`

Definition at line 294 of file `arrays.c`.

References `ar_alloc()`, `ar_avail_get()`, `ar_length_get()`, `ar_length_set()`, `ARRAY_TYPE_EMPTY`, `log_pedantic`, `mm_free()`, and `mm_move()`.

Referenced by `ar_dupe()`, `imap_parse_arguments()`, `imap_parse_array()`, `imap_parse_dataitems()`, `mail_mime_part()`, `mail_mime_split()`, `mail_mime_type_parameters()`, `meta_data_fetch_message_tags()`, and `portal_smtp_create_data()`.

5.1.1.3 `size_t ar_avail_get (array_t *array)`

Get the maximum number of elements allocated in an array.

Parameters:

array a pointer to the array to be examined.

Returns:

NULL on failure, or the number of elements the array can hold.

Definition at line 52 of file `arrays.c`.

References `log_pedantic`.

Referenced by `ar_append()`, `ar_dupe()`, `ar_free()`, and `ar_length_set()`.

5.1.1.4 `array_t* ar_dupe (array_t *array)`

Create a duplicate copy of the array.

Note:

The new array might not correspond precisely to the layout of the original.
Deep copies will be made of all elements that are also arrays.. Managed strings will be copied, but neither empty strings nor pointers will be.

Parameters:

array a pointer to the array to be copied.

Returns:

NULL on failure, or a pointer to the new copy of the array on success.

Definition at line 411 of file arrays.c.

References `ar_alloc()`, `ar_append()`, `ar_avail_get()`, `ar_dupe()`, `ARRAY_TYPE_ARRAY`, `ARRAY_TYPE_EMPTY`, `ARRAY_TYPE_POINTER`, `ARRAY_TYPE_STRINGER`, `st_dupe()`, and `type()`.

Referenced by `ar_dupe()`, and `meta_message_dupe()`.

5.1.1.5 `array_t* ar_field_ar (array_t * array, size_t element)`

Return the value of an element in an array that holds another array.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a pointer to an array object with the element's data on success.

Definition at line 182 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_ARRAY`, and `log_pedantic`.

Referenced by `imap_fetch_body()`.

5.1.1.6 `chr_t* ar_field_ns (array_t * array, size_t element)`

Return the value of an element in an array that holds a null-terminated string.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a pointer to a null-terminated string with the element's data on success.

Definition at line 153 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_NULLER`, and `log_pedantic`.

5.1.1.7 `placer_t* ar_field_pl (array_t * array, size_t element)`

Return the value of an element in an array that holds a placer.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a pointer to a placer with the element's data on success.

Definition at line 211 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_PLACER`, `log_pedantic`, and `placer_t`.

5.1.1.8 void* ar_field_ptr (array_t * array, size_t element)

Return the value of an element in an array that holds a pointer.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a generic pointer to the element's data on success.

Definition at line 240 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_POINTER`, and `log_pedantic`.

Referenced by `imap_fetch_body_part()`, `imap_fetch_bodystructure()`, `mail_mime_free()`, `mail_mime_generate_boundary()`, `portal_message_attachments()`, and `portal_message_body()`.

5.1.1.9 stringer_t* ar_field_st (array_t * array, size_t element)

Return the value of an element in an array that holds a managed string.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a pointer to a managed string with the element's data on success.

Definition at line 124 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_STRINGER`, and `log_pedantic`.

Referenced by `imap_fetch_bodystructure()`, `mail_mime_part()`, `meta_folders_stats_tags()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, and `portal_message_tags_array()`.

5.1.1.10 uint32_t ar_field_type (array_t * array, size_t element)

Definition at line 93 of file arrays.c.

References `ar_length_get()`, `ARRAY_TYPE_EMPTY`, and `log_pedantic`.

Referenced by `ar_field_ar()`, `ar_field_ns()`, `ar_field_pl()`, `ar_field_ptr()`, `ar_field_st()`, and `imap_fetch_body()`.

5.1.1.11 void ar_free (array_t * array)

Free an array object and all of its underlying elements.

Note:

Array elements that are managed strings, and aren't empty or of `ARRAY_TYPE_POINTER` will be freed.

Returns:

This function returns no value.

Definition at line 368 of file arrays.c.

References `ar_avail_get()`, `ar_free()`, `ARRAY_TYPE_ARRAY`, `ARRAY_TYPE_EMPTY`, `ARRAY_TYPE_PLACER`, `ARRAY_TYPE_POINTER`, `ARRAY_TYPE_STRINGER`, `log_pedantic`, `mm_free()`, `st_free()`, and `type()`.

Referenced by `ar_free()`, `imap_command_parser()`, `imap_fetch_bodystructure()`, `imap_parse_arguments()`, `imap_parse_array()`, `imap_session_destroy()`, `mail_mime_free()`, `mail_mime_part()`, `mail_mime_type_parameters()`, `meta_message_free()`, `portal_endpoint_messages_tag()`, and `portal_smtp_create_data()`.

5.1.1.12 `size_t ar_length_get (array_t * array)`

Get the number of elements actively stored in an array.

Parameters:

array a pointer to the array to be counted.

Returns:

NULL on failure, or the number of elements currently held in the array.

Definition at line 71 of file arrays.c.

References `log_pedantic`.

Referenced by `ar_append()`, `ar_field_ar()`, `ar_field_ns()`, `ar_field_pl()`, `ar_field_ptr()`, `ar_field_st()`, `ar_field_type()`, `imap_append()`, `imap_close()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_fetch_body_tag()`, `imap_fetch_bodystructure()`, `imap_flag_parse()`, `imap_get_ar_ar()`, `imap_get_ptr()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_id()`, `imap_list()`, `imap_login()`, `imap_lsub()`, `imap_parse_dataitems()`, `imap_rename()`, `imap_search()`, `imap_search_messages_inner()`, `imap_select()`, `imap_status()`, `imap_store()`, `imap_subscribe()`, `mail_mime_free()`, `mail_mime_generate_boundary()`, `mail_mime_part()`, `mail_mime_type_parameters()`, `meta_folders_stats_tags()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_message_attachments()`, `portal_message_tags_array()`, and `portal_smtp_create_data()`.

5.1.1.13 `void ar_length_set (array_t * array, size_t used)`

Manually set the number of used elements in an array.

Parameters:

array a pointer to the array to be adjusted.

used the new number of elements (less than the array's maximum size) to be set as the array's used length.

Returns:

This function returns no value.

Definition at line 269 of file arrays.c.

References `ar_avail_get()`, and `log_pedantic`.

Referenced by `ar_append()`.

5.2 src/core/buckets/buckets.h File Reference

Defines

- `#define MAGMA_CORE_POOL_OBJECTS_LIMIT` 4096
- `#define MAGMA_CORE_POOL_TIMEOUT_LIMIT` 86400
- `#define ARRAY_MAX_ELEMENTS` 16384
- `#define ARRAY_TYPE_EMPTY` 0
- `#define ARRAY_TYPE_ARRAY` 1
- `#define ARRAY_TYPE_STRINGER` 2
- `#define ARRAY_TYPE_SIZER` 3
- `#define ARRAY_TYPE_NULLER` 4
- `#define ARRAY_TYPE_PLACER` 5
- `#define ARRAY_TYPE_POINTER` 6

Typedefs

- `typedef char array_t`
- `typedef M_POOL_STATUS status_t`

Enumerations

- `enum M_POOL_STATUS { PL_ERROR = -1, PL_AVAILABLE = 0, PL_RESERVED = 1 }`

Functions

- `struct __attribute__((packed))`
- `void pool_free (pool_t *pool)`
Free an object pool.
- `uint32_t pool_get_count (pool_t *pool)`
Return the total number of items allocated inside a pool.
- `uint32_t pool_get_timeout (pool_t *pool)`
Get the timeout value for a pool.
- `uint64_t pool_get_failures (pool_t *pool)`
Get the number of failed requests for a pool.
- `uint32_t pool_get_available (pool_t *pool)`
Return the total number of items actively available in a pool.
- `pool_t * pool_alloc (uint32_t count, uint32_t timeout)`
Allocate a new object pool.
- `status_t pool_get_status (pool_t *pool, uint32_t item)`
Get the status flag for an item in a pool.
- `status_t pool_set_status (pool_t *pool, uint32_t item, status_t status)`
Set the status flag for an item in a pool.
- `void pool_release (pool_t *pool, uint32_t item)`

Return an object to a pool and set its status to PL_AVAILABLE.

- void * [pool_get_obj](#) (pool_t *pool, uint32_t item)

Return a specified object from a pool.

- status_t [pool_pull](#) (pool_t *pool, uint32_t *item)

Return the first available object in a pool.

- void * [pool_swap_obj](#) (pool_t *pool, uint32_t item, void *object)

Wait to acquire the value of an object in a pool, and return the original value while updating it with a new value.

- void * [pool_set_obj](#) (pool_t *pool, uint32_t item, void *object)

Set the object pointer for a given item in a pool.

- array_t * [ar_alloc](#) (size_t size)

arrays.c

- int_t [ar_append](#) (array_t **array, uint32_t type, void *item)

- size_t [ar_avail_get](#) (array_t *array)

Get the maximum number of elements allocated in an array.

- array_t * [ar_dupe](#) (array_t *array)

Create a duplicate copy of the array.

- array_t * [ar_field_ar](#) (array_t *array, size_t element)

Return the value of an element in an array that holds another array.

- chr_t * [ar_field_ns](#) (array_t *array, size_t element)

Return the value of an element in an array that holds a null-terminated string.

- placer_t * [ar_field_pl](#) (array_t *array, size_t element)

Return the value of an element in an array that holds a placer.

- void * [ar_field_ptr](#) (array_t *array, size_t element)

Return the value of an element in an array that holds a pointer.

- stringer_t * [ar_field_st](#) (array_t *array, size_t element)

Return the value of an element in an array that holds a managed string.

- uint32_t [ar_field_type](#) (array_t *array, size_t element)

- void [ar_free](#) (array_t *array)

Free an array object and all of its underlying elements.

- size_t [ar_length_get](#) (array_t *array)

Get the number of elements actively stored in an array.

- void [ar_length_set](#) (array_t *array, size_t used)

Manually set the number of used elements in an array.

- int_t [stacker_push](#) (stacker_t *stack, void *data)

stacked.c

- stacker_t * [stacker_alloc](#) (void *free_function)

Allocate a new instance of a stacked list.

- unsigned long [stacker_nodes](#) ([stacker_t](#) *stack)

Get the number of nodes in a stacked list.

- void * [stacker_pop](#) ([stacker_t](#) *stack)

Pop the last entry off a stacked list.

- void [stacker_free](#) ([stacker_t](#) *stack)

Free a stacked list and all of its underlying data nodes.

Variables

- [stacker_node_t](#)
- [stacker_t](#)
- [pool_t](#)

5.2.1 Define Documentation

5.2.1.1 #define ARRAY_MAX_ELEMENTS 16384

Definition at line 26 of file buckets.h.

Referenced by [ar_alloc\(\)](#).

5.2.1.2 #define ARRAY_TYPE_ARRAY 1

Definition at line 28 of file buckets.h.

Referenced by [ar_dupe\(\)](#), [ar_field_ar\(\)](#), [ar_free\(\)](#), [imap_fetch_body\(\)](#), and [imap_parse_dataitems\(\)](#).

5.2.1.3 #define ARRAY_TYPE_EMPTY 0

Definition at line 27 of file buckets.h.

Referenced by [ar_append\(\)](#), [ar_dupe\(\)](#), [ar_field_type\(\)](#), and [ar_free\(\)](#).

5.2.1.4 #define ARRAY_TYPE_NULLER 4

Definition at line 31 of file buckets.h.

Referenced by [ar_field_ns\(\)](#).

5.2.1.5 #define ARRAY_TYPE_PLACER 5

Definition at line 32 of file buckets.h.

Referenced by [ar_field_pl\(\)](#), and [ar_free\(\)](#).

5.2.1.6 #define ARRAY_TYPE_POINTER 6

Definition at line 33 of file buckets.h.

Referenced by [ar_dupe\(\)](#), [ar_field_ptr\(\)](#), [ar_free\(\)](#), [imap_parse_dataitems\(\)](#), [mail_mime_part\(\)](#), and [portal_smtp_create_data\(\)](#).

5.2.1.7 #define ARRAY_TYPE_SIZER 3

Definition at line 30 of file buckets.h.

5.2.1.8 #define ARRAY_TYPE_STRINGER 2

Definition at line 29 of file buckets.h.

Referenced by ar_dupe(), ar_field_st(), ar_free(), mail_mime_split(), mail_mime_type_parameters(), and meta_data_fetch_message_tags().

5.2.1.9 #define MAGMA_CORE_POOL_OBJECTS_LIMIT 4096

The maximum number of objects allowed inside a pool.

Definition at line 18 of file buckets.h.

Referenced by pool_alloc(), and sanity_check().

5.2.1.10 #define MAGMA_CORE_POOL_TIMEOUT_LIMIT 86400

The maximum number of seconds a pool can be configured to wait in seconds.

Definition at line 23 of file buckets.h.

Referenced by pool_alloc(), and sanity_check().

5.2.2 Typedef Documentation

5.2.2.1 typedef char array_t

Definition at line 36 of file buckets.h.

5.2.2.2 typedef M_POOL_STATUS status_t

Definition at line 44 of file buckets.h.

5.2.3 Enumeration Type Documentation

5.2.3.1 enum M_POOL_STATUS

Enumerator:

PL_ERROR

PL_AVAILABLE

PL_RESERVED

Definition at line 38 of file buckets.h.

5.2.4 Function Documentation

5.2.4.1 struct __attribute__((packed)) [read]

< The type of the rr set covered by this record.

- < The numerical id of the cryptographic algorithm used to generate the signature.
- < The number of labels in the original RRSIG RR owner name.
- < The original TTL of the covered rrset as it appears in the authoritative zone.
- < The expiration date after which the RRSIG record must not be used for authentication.
- < The inception date before which the RRSIG record must not be used for authentication.
- < A tag (selector) identifying the corresponding DNSKEY RR (which isn't necessarily unique) that validates this record.
- < A label identifying the owner name of the DNSKEY RR validating this signature (must not use name compression).

Definition at line 58 of file buckets.h.

References count, lock, objects, and status.

5.2.4.2 `array_t* ar_alloc (size_t size)`

[arrays.c](#) [arrays.c](#)

Parameters:

size the number of elements the array is capable of holding.

Returns:

NULL on failure, or a pointer to the newly allocated array on success.

Definition at line 15 of file arrays.c.

References ARRAY_MAX_ELEMENTS, log_error, log_pedantic, and mm_wipe().

Referenced by ar_append(), ar_dupe(), mail_mime_part(), mail_mime_split(), mail_mime_type_parameters(), and meta_data_fetch_message_tags().

5.2.4.3 `int_t ar_append (array_t ** array, uint32_t type, void * item)`

Definition at line 294 of file arrays.c.

References ar_alloc(), ar_avail_get(), ar_length_get(), ar_length_set(), ARRAY_TYPE_EMPTY, log_pedantic, mm_free(), and mm_move().

Referenced by ar_dupe(), imap_parse_arguments(), imap_parse_array(), imap_parse_dataitems(), mail_mime_part(), mail_mime_split(), mail_mime_type_parameters(), meta_data_fetch_message_tags(), and portal_smtp_create_data().

5.2.4.4 `size_t ar_avail_get (array_t * array)`

Get the maximum number of elements allocated in an array.

Parameters:

array a pointer to the array to be examined.

Returns:

NULL on failure, or the number of elements the array can hold.

Definition at line 52 of file arrays.c.

References log_pedantic.

Referenced by ar_append(), ar_dupe(), ar_free(), and ar_length_set().

5.2.4.5 `array_t* ar_dupe (array_t * array)`

Create a duplicate copy of the array.

Note:

The new array might not correspond precisely to the layout of the original.

Deep copies will be made of all elements that are also arrays.. Managed strings will be copied, but neither empty strings nor pointers will be.

Parameters:

array a pointer to the array to be copied.

Returns:

NULL on failure, or a pointer to the new copy of the array on success.

Definition at line 411 of file arrays.c.

References `ar_alloc()`, `ar_append()`, `ar_avail_get()`, `ar_dupe()`, `ARRAY_TYPE_ARRAY`, `ARRAY_TYPE_EMPTY`, `ARRAY_TYPE_POINTER`, `ARRAY_TYPE_STRINGER`, `st_dupe()`, and `type()`.

Referenced by `ar_dupe()`, and `meta_message_dupe()`.

5.2.4.6 `array_t* ar_field_ar (array_t * array, size_t element)`

Return the value of an element in an array that holds another array.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a pointer to an array object with the element's data on success.

Definition at line 182 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_ARRAY`, and `log_pedantic`.

Referenced by `imap_fetch_body()`.

5.2.4.7 `chr_t* ar_field_ns (array_t * array, size_t element)`

Return the value of an element in an array that holds a null-terminated string.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a pointer to a null-terminated string with the element's data on success.

Definition at line 153 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_NULLER`, and `log_pedantic`.

5.2.4.8 `placer_t* ar_field_pl (array_t * array, size_t element)`

Return the value of an element in an array that holds a placer.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a pointer to a placer with the element's data on success.

Definition at line 211 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_PLACER`, `log_pedantic`, and `placer_t`.

5.2.4.9 `void* ar_field_ptr (array_t * array, size_t element)`

Return the value of an element in an array that holds a pointer.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a generic pointer to the element's data on success.

Definition at line 240 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_POINTER`, and `log_pedantic`.

Referenced by `imap_fetch_body_part()`, `imap_fetch_bodystructure()`, `mail_mime_free()`, `mail_mime_generate_boundary()`, `portal_message_attachments()`, and `portal_message_body()`.

5.2.4.10 `stringer_t* ar_field_st (array_t * array, size_t element)`

Return the value of an element in an array that holds a managed string.

Parameters:

array a pointer to the array to be examined. *element* the index of the element in the array to be queried.

Returns:

NULL on failure, or a pointer to a managed string with the element's data on success.

Definition at line 124 of file arrays.c.

References `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_STRINGER`, and `log_pedantic`.

Referenced by `imap_fetch_bodystructure()`, `mail_mime_part()`, `meta_folders_stats_tags()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, and `portal_message_tags_array()`.

5.2.4.11 `uint32_t ar_field_type (array_t * array, size_t element)`

Definition at line 93 of file arrays.c.

References `ar_length_get()`, `ARRAY_TYPE_EMPTY`, and `log_pedantic`.

Referenced by `ar_field_ar()`, `ar_field_ns()`, `ar_field_pl()`, `ar_field_ptr()`, `ar_field_st()`, and `imap_fetch_body()`.

5.2.4.12 void ar_free (array_t * array)

Free an array object and all of its underlying elements.

Note:

Array elements that are managed strings, and aren't empty or of ARRAY_TYPE_POINTER will be freed.

Returns:

This function returns no value.

Definition at line 368 of file arrays.c.

References ar_avail_get(), ar_free(), ARRAY_TYPE_ARRAY, ARRAY_TYPE_EMPTY, ARRAY_TYPE_PLACER, ARRAY_TYPE_POINTER, ARRAY_TYPE_STRINGER, log_pedantic, mm_free(), st_free(), and type().

Referenced by ar_free(), imap_command_parser(), imap_fetch_bodystructure(), imap_parse_arguments(), imap_parse_array(), imap_session_destroy(), mail_mime_free(), mail_mime_part(), mail_mime_type_parameters(), meta_message_free(), portal_endpoint_messages_tag(), and portal_smtp_create_data().

5.2.4.13 size_t ar_length_get (array_t * array)

Get the number of elements actively stored in an array.

Parameters:

array a pointer to the array to be counted.

Returns:

NULL on failure, or the number of elements currently held in the array.

Definition at line 71 of file arrays.c.

References log_pedantic.

Referenced by ar_append(), ar_field_ar(), ar_field_ns(), ar_field_pl(), ar_field_ptr(), ar_field_st(), ar_field_type(), imap_append(), imap_close(), imap_copy(), imap_create(), imap_delete(), imap_examine(), imap_expunge(), imap_fetch(), imap_fetch_body(), imap_fetch_body_header(), imap_fetch_body_tag(), imap_fetch_bodystructure(), imap_flag_parse(), imap_get_ar_ar(), imap_get_ptr(), imap_get_st_ar(), imap_get_type_ar(), imap_id(), imap_list(), imap_login(), imap_lsub(), imap_parse_dataitems(), imap_rename(), imap_search(), imap_search_messages_inner(), imap_select(), imap_status(), imap_store(), imap_subscribe(), mail_mime_free(), mail_mime_generate_boundary(), mail_mime_part(), mail_mime_type_parameters(), meta_folders_stats_tags(), portal_endpoint_messages_list(), portal_endpoint_messages_tag(), portal_message_attachments(), portal_message_tags_array(), and portal_smtp_create_data().

5.2.4.14 void ar_length_set (array_t * array, size_t used)

Manually set the number of used elements in an array.

Parameters:

array a pointer to the array to be adjusted.

used the new number of elements (less than the array's maximum size) to be set as the array's used length.

Returns:

This function returns no value.

Definition at line 269 of file arrays.c.

References ar_avail_get(), and log_pedantic.

Referenced by ar_append().

5.2.4.15 `pool_t* pool_alloc (uint32_t count, uint32_t timeout)`

Allocate a new object pool.

Parameters:

count the number of items the pool can hold.

timeout a timeout for pool requests in seconds (specify 0 for infinite wait).

Returns:

NULL on failure,

Definition at line 33 of file pool.c.

References `log_info`, `MAGMA_CORE_POOL_OBJECTS_LIMIT`, `MAGMA_CORE_POOL_TIMEOUT_LIMIT`, `mm_alloc()`, `mm_free()`, `mutex_init()`, and `pool_t`.

Referenced by `cache_start()`, `spf_start()`, and `sql_start()`.

5.2.4.16 `void pool_free (pool_t * pool)`

Free an object pool.

Warning:

This function will not free the underlying objects contained by the pool!

Parameters:

pool the pool to be released from memory.

Returns:

This function returns no value.

Definition at line 16 of file pool.c.

References `mm_free()`, and `mutex_destroy()`.

Referenced by `cache_stop()`, `spf_stop()`, and `sql_stop()`.

5.2.4.17 `uint32_t pool_get_available (pool_t * pool)`

Return the total number of items actively available in a pool.

Parameters:

pool the pool to be queried.

Returns:

0 on failure, or the total number of items actively available in the specified pool.

Definition at line 94 of file pool.c.

5.2.4.18 uint32_t pool_get_count (pool_t * pool)

Return the total number of items allocated inside a pool.

Note:

Some of the item slots may be unused, but remain reserved.

Parameters:

pool the pool to be queried.

Returns:

0 on failure, or the allocation count of the pool on success.

Definition at line 83 of file pool.c.

Referenced by pool_get_obj(), and pool_set_obj().

5.2.4.19 uint64_t pool_get_failures (pool_t * pool)

Get the number of failed requests for a pool.

Note:

Most of the time, a failure will correspond to a timeout, but can also be triggered by other error conditions.

Parameters:

pool a pointer to the pool to be examined.

Returns:

the number of failed requests made on the specified pool.

Definition at line 119 of file pool.c.

References mutex_lock(), and mutex_unlock().

5.2.4.20 void* pool_get_obj (pool_t * pool, uint32_t item)

Return a specified object from a pool.

Parameters:

pool the pool containing the object.

item the identifier of the object to query.

Returns:

NULL on failure, or the object corresponding to the specified item number.

Definition at line 241 of file pool.c.

References log_pedantic, mutex_lock(), mutex_unlock(), and pool_get_count().

Referenced by cache_add(), cache_append(), cache_decrement(), cache_delete(), cache_flush(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_silent_add(), cache_stop(), dspam_check(), dspam_train(), mail_db_update_message_folder(), pool_swap_obj(), spf_check(), spf_stop(), sql_insert_conn(), sql_num_rows_conn(), sql_ping(), sql_query_conn(), sql_query_res_conn(), sql_stop(), sql_write_conn(), stmt_rebuild(), stmt_start(), tran_commit(), tran_rollback(), and tran_start().

5.2.4.21 status_t pool_get_status (pool_t * pool, uint32_t item)

Get the status flag for an item in a pool.

Note:

A value of PL_AVAILABLE indicates the object is available for use;; PL_RESERVED indicates the object is in use by a worker thread.

Parameters:

pool the pool containing the specified item.

item the identifier of the item to be queried.

Returns:

PL_ERROR on failure; otherwise, the status of the specified item.

Definition at line 139 of file pool.c.

References log_check, log_pedantic, PL_AVAILABLE, PL_ERROR, and PL_RESERVED.

Referenced by pool_pull(), and pool_swap_obj().

5.2.4.22 uint32_t pool_get_timeout (pool_t * pool)

Get the timeout value for a pool.

Note:

A timeout of zero will cause threads to wait forever.

Parameters:

pool a pointer to the pool to be examined.

Returns:

the timeout value of the specified pool, in seconds.

Definition at line 107 of file pool.c.

5.2.4.23 status_t pool_pull (pool_t * pool, uint32_t * item)

Return the first available object in a pool.

Note:

If no object can be returned immediately, wait for the pool's configured timeout value, in seconds, for an object to become available. If the timeout is zero, wait indefinitely.

Parameters:

item A pointer to a number that will store the zero-based indexed of the first available item in the pool.

Returns:

PL_RESERVED on success or PL_ERROR if an object couldn't be reserved.

Definition at line 178 of file pool.c.

References mutex_lock(), mutex_unlock(), PL_AVAILABLE, PL_ERROR, PL_RESERVED, pool_get_status(), and pool_set_status().

Referenced by cache_add(), cache_append(), cache_decrement(), cache_delete(), cache_flush(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_silent_add(), dspam_check(), dspam_train(), spf_check(), sql_insert(), sql_num_rows(), sql_query(), sql_query_res(), sql_write(), stmt_exec(), stmt_exec_affected(), stmt_get_result(), stmt_insert(), and tran_start().

5.2.4.24 void pool_release (pool_t * pool, uint32_t item)

Return an object to a pool and set its status to PL_AVAILABLE.

Parameters:

pool the pool tracking the returned item.

item the identifier of the item being returned.

Returns:

This function returns no value.

Definition at line 226 of file pool.c.

References mutex_lock(), mutex_unlock(), PL_AVAILABLE, and pool_set_status().

Referenced by cache_add(), cache_append(), cache_decrement(), cache_delete(), cache_flush(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_silent_add(), dspam_check(), dspam_train(), spf_check(), sql_insert(), sql_num_rows(), sql_query(), sql_query_res(), sql_write(), stmt_exec(), stmt_exec_affected(), stmt_get_result(), stmt_insert(), tran_commit(), tran_rollback(), and tran_start().

5.2.4.25 void* pool_set_obj (pool_t * pool, uint32_t item, void * object)

Set the object pointer for a given item in a pool.

Parameters:

pool the pool containing the specified item.

item the item id to be adjusted.

object the new value of the object corresponding to the specified item.

Returns:

NULL on failure, or a pointer to the object that was set.

Definition at line 267 of file pool.c.

References log_pedantic, mutex_lock(), mutex_unlock(), and pool_get_count().

Referenced by cache_start(), pool_swap_obj(), spf_start(), and sql_start().

5.2.4.26 status_t pool_set_status (pool_t * pool, uint32_t item, status_t status)

Set the status flag for an item in a pool.

Note:

A value of PL_AVAILABLE indicates the object is available for use,; PL_RESERVED indicates the object is in use by a worker thread.

Parameters:

pool the pool containing the specified item.

item the identifier of the item to be adjusted.

status the new status for the item.

Returns:

PL_ERROR on failure; otherwise, the new status value of the specified item.

Definition at line 159 of file pool.c.

References log_check, log_pedantic, PL_AVAILABLE, PL_ERROR, and PL_RESERVED.

Referenced by pool_pull(), and pool_release().

5.2.4.27 void* pool_swap_obj (pool_t * pool, uint32_t item, void * object)

Wait to acquire the value of an object in a pool, and return the original value while updating it with a new value.

Parameters:

pool a pointer to the pool to be modified.

item the zero-based index of the object in the pool to be updated.

object the new value of the object corresponding to the specified item.

Returns:

a pointer to the original pool object's value, or NULL on failure.

LOW: Currently the function loops until the requested object is available. A superior implementation would hook into the release function and detect when the desired object is available and perform the swap at that point.

Definition at line 293 of file pool.c.

References mutex_lock(), mutex_unlock(), PL_AVAILABLE, pool_get_obj(), pool_get_status(), and pool_set_obj().

5.2.4.28 stacker_t* stacker_alloc (void * free_function)

Allocate a new instance of a stacked list.

Parameters:

free_function if not NULL, a pointer to a function that will be used to free the data underlying each node in the stacked list.

Returns:

NULL on failure, or a pointer to the newly created stack list on success.

Definition at line 46 of file stacked.c.

References log_pedantic, mm_alloc(), mm_free(), mutex_init(), and stacker_t.

5.2.4.29 void stacker_free (stacker_t * stack)

Free a stacked list and all of its underlying data nodes.

Parameters:

stack a pointer to the stacked list to be freed.

Returns:

This function returns no value.

Definition at line 15 of file stacked.c.

References mm_free(), mutex_destroy(), and stacker_node_t.

5.2.4.30 unsigned long stacker_nodes (stacker_t * *stack*)

Get the number of nodes in a stacked list.

Parameters:

stack a pointer to the stacked list to be queried.

Returns:

the number of nodes currently held by the specified stacked list.

Definition at line 75 of file stacked.c.

References mutex_lock(), and mutex_unlock().

5.2.4.31 void* stacker_pop (stacker_t * *stack*)

Pop the last entry off a stacked list.

Parameters:

stack a pointer to the stacked list to be queried.

Returns:

NULL on failure or the value of the last node in the stacked list.

Definition at line 141 of file stacked.c.

References mm_free(), mutex_lock(), mutex_unlock(), and stacker_node_t.

5.2.4.32 int_t stacker_push (stacker_t * *stack*, void * *data*)

[stacked.c](#) [stacked.c](#)

Parameters:

stack a pointer to the stacked list to be updated.

data a pointer to the data to associated with the new stacked list node.

Returns:

0 on failure or 1 on success.

Definition at line 96 of file stacked.c.

References log_error, log_pedantic, mm_alloc(), mutex_lock(), mutex_unlock(), and stacker_node_t.

5.2.5 Variable Documentation

5.2.5.1 pool_t

Definition at line 66 of file buckets.h.

Referenced by pool_alloc().

5.2.5.2 `stacker_node_t`

Definition at line 49 of file buckets.h.

Referenced by `stacker_free()`, `stacker_pop()`, and `stacker_push()`.

5.2.5.3 `stacker_t`

Definition at line 56 of file buckets.h.

Referenced by `stacker_alloc()`.

5.3 src/core/buckets/pool.c File Reference

```
#include "magma.h"
```

Functions

- void `pool_free` (`pool_t` *pool)
Free an object pool.
- `pool_t` * `pool_alloc` (`uint32_t` count, `uint32_t` timeout)
Allocate a new object pool.
- `uint32_t` `pool_get_count` (`pool_t` *pool)
Return the total number of items allocated inside a pool.
- `uint32_t` `pool_get_available` (`pool_t` *pool)
Return the total number of items actively available in a pool.
- `uint32_t` `pool_get_timeout` (`pool_t` *pool)
Get the timeout value for a pool.
- `uint64_t` `pool_get_failures` (`pool_t` *pool)
Get the number of failed requests for a pool.
- `status_t` `pool_get_status` (`pool_t` *pool, `uint32_t` item)
Get the status flag for an item in a pool.
- `status_t` `pool_set_status` (`pool_t` *pool, `uint32_t` item, `status_t` status)
Set the status flag for an item in a pool.
- `status_t` `pool_pull` (`pool_t` *pool, `uint32_t` *item)
Return the first available object in a pool.
- void `pool_release` (`pool_t` *pool, `uint32_t` item)
Return an object to a pool and set its status to PL_AVAILABLE.
- void * `pool_get_obj` (`pool_t` *pool, `uint32_t` item)
Return a specified object from a pool.
- void * `pool_set_obj` (`pool_t` *pool, `uint32_t` item, void *object)
Set the object pointer for a given item in a pool.
- void * `pool_swap_obj` (`pool_t` *pool, `uint32_t` item, void *object)
Wait to acquire the value of an object in a pool, and return the original value while updating it with a new value.

5.3.1 Function Documentation

5.3.1.1 `pool_t`* `pool_alloc` (`uint32_t` count, `uint32_t` timeout)

Allocate a new object pool.

Parameters:

count the number of items the pool can hold.

timeout a timeout for pool requests in seconds (specify 0 for infinite wait).

Returns:

NULL on failure,

Definition at line 33 of file pool.c.

References log_info, MAGMA_CORE_POOL_OBJECTS_LIMIT, MAGMA_CORE_POOL_TIMEOUT_LIMIT, mm_alloc(), mm_free(), mutex_init(), and pool_t.

Referenced by cache_start(), spf_start(), and sql_start().

5.3.1.2 void pool_free (pool_t * pool)

Free an object pool.

Warning:

This function will not free the underlying objects contained by the pool!

Parameters:

pool the pool to be released from memory.

Returns:

This function returns no value.

Definition at line 16 of file pool.c.

References mm_free(), and mutex_destroy().

Referenced by cache_stop(), spf_stop(), and sql_stop().

5.3.1.3 uint32_t pool_get_available (pool_t * pool)

Return the total number of items actively available in a pool.

Parameters:

pool the pool to be queried.

Returns:

0 on failure, or the total number of items actively available in the specified pool.

Definition at line 94 of file pool.c.

5.3.1.4 uint32_t pool_get_count (pool_t * pool)

Return the total number of items allocated inside a pool.

Note:

Some of the item slots may be unused, but remain reserved.

Parameters:

pool the pool to be queried.

Returns:

0 on failure, or the allocation count of the pool on success.

Definition at line 83 of file pool.c.

Referenced by pool_get_obj(), and pool_set_obj().

5.3.1.5 uint64_t pool_get_failures (pool_t * pool)

Get the number of failed requests for a pool.

Note:

Most of the time, a failure will correspond to a timeout, but can also be triggered by other error conditions.

Parameters:

pool a pointer to the pool to be examined.

Returns:

the number of failed requests made on the specified pool.

Definition at line 119 of file pool.c.

References mutex_lock(), and mutex_unlock().

5.3.1.6 void* pool_get_obj (pool_t * pool, uint32_t item)

Return a specified object from a pool.

Parameters:

pool the pool containing the object.

item the identifier of the object to query.

Returns:

NULL on failure, or the object corresponding to the specified item number.

Definition at line 241 of file pool.c.

References log_pedantic, mutex_lock(), mutex_unlock(), and pool_get_count().

Referenced by cache_add(), cache_append(), cache_decrement(), cache_delete(), cache_flush(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_silent_add(), cache_stop(), dspam_check(), dspam_train(), mail_db_update_message_folder(), pool_swap_obj(), spf_check(), spf_stop(), sql_insert_conn(), sql_num_rows_conn(), sql_ping(), sql_query_conn(), sql_query_res_conn(), sql_stop(), sql_write_conn(), stmt_rebuild(), stmt_start(), tran_commit(), tran_rollback(), and tran_start().

5.3.1.7 status_t pool_get_status (pool_t * pool, uint32_t item)

Get the status flag for an item in a pool.

Note:

A value of PL_AVAILABLE indicates the object is available for use;; PL_RESERVED indicates the object is in use by a worker thread.

Parameters:

pool the pool containing the specified item.

item the identifier of the item to be queried.

Returns:

PL_ERROR on failure; otherwise, the status of the specified item.

Definition at line 139 of file pool.c.

References log_check, log_pedantic, PL_AVAILABLE, PL_ERROR, and PL_RESERVED.

Referenced by pool_pull(), and pool_swap_obj().

5.3.1.8 uint32_t pool_get_timeout (pool_t * pool)

Get the timeout value for a pool.

Note:

A timeout of zero will cause threads to wait forever.

Parameters:

pool a pointer to the pool to be examined.

Returns:

the timeout value of the specified pool, in seconds.

Definition at line 107 of file pool.c.

5.3.1.9 status_t pool_pull (pool_t * pool, uint32_t * item)

Return the first available object in a pool.

Note:

If no object can be returned immediately, wait for the pool's configured timeout value, in seconds, for an object to become available. If the timeout is zero, wait indefinitely.

Parameters:

item A pointer to a number that will store the zero-based indexed of the first available item in the pool.

Returns:

PL_RESERVED on success or PL_ERROR if an object couldn't be reserved.

Definition at line 178 of file pool.c.

References mutex_lock(), mutex_unlock(), PL_AVAILABLE, PL_ERROR, PL_RESERVED, pool_get_status(), and pool_set_status().

Referenced by cache_add(), cache_append(), cache_decrement(), cache_delete(), cache_flush(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_silent_add(), dspam_check(), dspam_train(), spf_check(), sql_insert(), sql_num_rows(), sql_query(), sql_query_res(), sql_write(), stmt_exec(), stmt_exec_affected(), stmt_get_result(), stmt_insert(), and tran_start().

5.3.1.10 void pool_release (pool_t * pool, uint32_t item)

Return an object to a pool and set its status to PL_AVAILABLE.

Parameters:

pool the pool tracking the returned item.

item the identifier of the item being returned.

Returns:

This function returns no value.

Definition at line 226 of file pool.c.

References mutex_lock(), mutex_unlock(), PL_AVAILABLE, and pool_set_status().

Referenced by cache_add(), cache_append(), cache_decrement(), cache_delete(), cache_flush(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_silent_add(), dspam_check(), dspam_train(), spf_check(), sql_insert(), sql_num_rows(), sql_query(), sql_query_res(), sql_write(), stmt_exec(), stmt_exec_affected(), stmt_get_result(), stmt_insert(), tran_commit(), tran_rollback(), and tran_start().

5.3.1.11 void* pool_set_obj (pool_t * pool, uint32_t item, void * object)

Set the object pointer for a given item in a pool.

Parameters:

pool the pool containing the specified item.

item the item id to be adjusted.

object the new value of the object corresponding to the specified item.

Returns:

NULL on failure, or a pointer to the object that was set.

Definition at line 267 of file pool.c.

References log_pedantic, mutex_lock(), mutex_unlock(), and pool_get_count().

Referenced by cache_start(), pool_swap_obj(), spf_start(), and sql_start().

5.3.1.12 status_t pool_set_status (pool_t * pool, uint32_t item, status_t status)

Set the status flag for an item in a pool.

Note:

A value of PL_AVAILABLE indicates the object is available for use,; PL_RESERVED indicates the object is in use by a worker thread.

Parameters:

pool the pool containing the specified item.

item the identifier of the item to be adjusted.

status the new status for the item.

Returns:

PL_ERROR on failure; otherwise, the new status value of the specified item.

Definition at line 159 of file pool.c.

References log_check, log_pedantic, PL_AVAILABLE, PL_ERROR, and PL_RESERVED.

Referenced by pool_pull(), and pool_release().

5.3.1.13 void* pool_swap_obj (pool_t * *pool*, uint32_t *item*, void * *object*)

Wait to acquire the value of an object in a pool, and return the original value while updating it with a new value.

Parameters:

pool a pointer to the pool to be modified.

item the zero-based index of the object in the pool to be updated.

object the new value of the object corresponding to the specified item.

Returns:

a pointer to the original pool object's value, or NULL on failure.

LOW: Currently the function loops until the requested object is available. A superior implementation would hook into the release function and detect when the desired object is available and perform the swap at that point.

Definition at line 293 of file pool.c.

References mutex_lock(), mutex_unlock(), PL_AVAILABLE, pool_get_obj(), pool_get_status(), and pool_set_obj().

5.4 src/core/buckets/stacked.c File Reference

```
#include "magma.h"
```

Functions

- void `stacker_free` (`stacker_t` *stack)
Free a stacked list and all of its underlying data nodes.
- `stacker_t` * `stacker_alloc` (void *free_function)
Allocate a new instance of a stacked list.
- uint64_t `stacker_nodes` (`stacker_t` *stack)
Get the number of nodes in a stacked list.
- int_t `stacker_push` (`stacker_t` *stack, void *data)
Push a new entry onto the end of a stacked list.
- void * `stacker_pop` (`stacker_t` *stack)
Pop the last entry off a stacked list.

5.4.1 Function Documentation

5.4.1.1 `stacker_t`* `stacker_alloc` (void * *free_function*)

Allocate a new instance of a stacked list.

Parameters:

free_function if not NULL, a pointer to a function that will be used to free the data underlying each node in the stacked list.

Returns:

NULL on failure, or a pointer to the newly created stack list on success.

Definition at line 46 of file stacked.c.

References `log_pedantic`, `mm_alloc()`, `mm_free()`, `mutex_init()`, and `stacker_t`.

5.4.1.2 void `stacker_free` (`stacker_t` * *stack*)

Free a stacked list and all of its underlying data nodes.

Parameters:

stack a pointer to the stacked list to be freed.

Returns:

This function returns no value.

Definition at line 15 of file stacked.c.

References `mm_free()`, `mutex_destroy()`, and `stacker_node_t`.

5.4.1.3 `uint64_t stacker_nodes (stacker_t * stack)`

Get the number of nodes in a stacked list.

Parameters:

stack a pointer to the stacked list to be queried.

Returns:

the number of nodes currently held by the specified stacked list.

Definition at line 75 of file `stacked.c`.

References `mutex_lock()`, and `mutex_unlock()`.

5.4.1.4 `void* stacker_pop (stacker_t * stack)`

Pop the last entry off a stacked list.

Parameters:

stack a pointer to the stacked list to be queried.

Returns:

NULL on failure or the value of the last node in the stacked list.

Definition at line 141 of file `stacked.c`.

References `mm_free()`, `mutex_lock()`, `mutex_unlock()`, and `stacker_node_t`.

5.4.1.5 `int_t stacker_push (stacker_t * stack, void * data)`

Push a new entry onto the end of a stacked list. [stacked.c](#)

Parameters:

stack a pointer to the stacked list to be updated.

data a pointer to the data to associated with the new stacked list node.

Returns:

0 on failure or 1 on success.

Definition at line 96 of file `stacked.c`.

References `log_error`, `log_pedantic`, `mm_alloc()`, `mutex_lock()`, `mutex_unlock()`, and `stacker_node_t`.

5.5 src/core/checksum/adler.c File Reference

```
#include "magma.h"
```

Functions

- `uint32_t hash_adler32 (void *buffer, size_t length)`

Return an Adler-32 hash of the specified data.

5.5.1 Function Documentation

5.5.1.1 `uint32_t hash_adler32 (void *buffer, size_t length)`

Return an Adler-32 hash of the specified data.

Parameters:

buffer a pointer to the data to be hashed.

length the length, in bytes, of the data to be hashed.

Returns:

a 32 bit number containing the Adler-32 hash of the data.

Definition at line 16 of file `adler.c`.

Referenced by `compress_bzip()`, `compress_import()`, `compress_lzo()`, `compress_zlib()`, `decompress_bzip()`, `decompress_lzo()`, `decompress_zlib()`, `deprecated_scramble_decrypt()`, `deprecated_scramble_encrypt()`, `deprecated_scramble_import()`, `scramble_decrypt()`, `scramble_encrypt()`, and `scramble_import()`.

5.6 src/core/checksum/checksum.h File Reference

Functions

- uint32_t [crc24_init](#) (void)
crc.c
- uint32_t [crc24_final](#) (uint32_t crc)
- uint32_t [crc24_checksum](#) (void *buffer, size_t length)
Generate a 24-bit CRC value for the Privacy Enhanced Message format.
- uint32_t [crc32_checksum](#) (void *buffer, size_t length)
Get 32-bit CRC value for a specified block of data.
- uint64_t [crc64_checksum](#) (void *buffer, size_t length)
Get 64-bit CRC value for a specified block of data.
- uint32_t [crc24_update](#) (void *buffer, size_t length, uint32_t crc)
- uint32_t [crc32_update](#) (void *buffer, size_t length, uint32_t crc)
Update a 32-bit CRC value with a check of additional data.
- uint64_t [crc64_update](#) (void *buffer, size_t length, uint64_t crc)
Update a 64-bit CRC value with a check of additional data.
- uint32_t [hash_adler32](#) (void *buffer, size_t length)
Return an Adler-32 hash of the specified data.
- uint32_t [hash_murmur32](#) (void *buffer, size_t length)
Generate a 32-bit Murmur hash of a block of data.
- uint64_t [hash_murmur64](#) (void *buffer, size_t length)
Generate a 64-bit Murmur hash of a block of data.
- uint32_t [hash_fletcher32](#) (void *buffer, size_t length)
Computer a 32-bit Fletcher hash for a block of data.

5.6.1 Function Documentation

5.6.1.1 uint32_t [crc24_checksum](#) (void * *buffer*, size_t *length*)

Generate a 24-bit CRC value for the Privacy Enhanced Message format.

Note:

This function uses the predefined initial value and polynomial defined in RFC 4880.

Parameters:

buffer a pointer to the data to be checked.

length the length, in bytes, of the input buffer.

Returns:

the updated 24-bit CRC value in a 32-bit unsigned integer.

Definition at line 59 of file crc.c.

References `crc24_final()`, `crc24_init()`, and `CRC24_POLY`.

Referenced by `prime_pem_unwrap()`, and `prime_pem_wrap()`.

5.6.1.2 `uint32_t crc24_final (uint32_t crc)`

Definition at line 48 of file crc.c.

Referenced by `crc24_checksum()`.

5.6.1.3 `uint32_t crc24_init (void)`

[crc.c](#)

Definition at line 26 of file crc.c.

References `CRC24_INIT`.

Referenced by `crc24_checksum()`.

5.6.1.4 `uint32_t crc24_update (void * buffer, size_t length, uint32_t crc)`

Definition at line 30 of file crc.c.

References `CRC24_POLY`.

5.6.1.5 `uint32_t crc32_checksum (void * buffer, size_t length)`

Get 32-bit CRC value for a specified block of data.

Parameters:

buffer a pointer to the data to be checked.

length the length, in bytes, of the input buffer.

Returns:

the 32-bit CRC value of the specified data.

Definition at line 115 of file crc.c.

References `crc32_update()`.

5.6.1.6 `uint32_t crc32_update (void * buffer, size_t length, uint32_t crc)`

Update a 32-bit CRC value with a check of additional data.

Parameters:

buffer a pointer to the data to be checked.

length the length, in bytes, of the input buffer.

crc the previously computed CRC value, or 0 if this is the initial pass.

Returns:

the updated 32-bit CRC value of the specified data.

Definition at line 83 of file crc.c.

References A, B, C, crc32_table, D, and S8.

Referenced by crc32_checksum().

5.6.1.7 uint64_t crc64_checksum (void * *buffer*, size_t *length*)

Get 64-bit CRC value for a specified block of data.

Parameters:

buffer a pointer to the data to be checked.

length the length, in bytes, of the input buffer.

Returns:

the 64-bit CRC value of the specified data.

Definition at line 156 of file crc.c.

References crc64_update().

Referenced by smtp_bounce(), and smtp_reply().

5.6.1.8 uint64_t crc64_update (void * *buffer*, size_t *length*, uint64_t *crc*)

Update a 64-bit CRC value with a check of additional data.

Parameters:

buffer a pointer to the data to be checked.

length the length, in bytes, of the input buffer.

crc the previously computed CRC value, or 0 if this is the initial pass.

Returns:

the updated 64-bit CRC value of the specified data.

Definition at line 126 of file crc.c.

References A, A1, B, C, crc64_table, D, S32, and S8.

Referenced by crc64_checksum().

5.6.1.9 uint32_t hash_adler32 (void * *buffer*, size_t *length*)

Return an Adler-32 hash of the specified data.

Parameters:

buffer a pointer to the data to be hashed.

length the length, in bytes, of the data to be hashed.

Returns:

a 32 bit number containing the Adler-32 hash of the data.

Definition at line 16 of file adler.c.

Referenced by compress_bzip(), compress_import(), compress_lzo(), compress_zlib(), decompress_bzip(), decompress_lzo(), decompress_zlib(), deprecated_scramble_decrypt(), deprecated_scramble_encrypt(), deprecated_scramble_import(), scramble_decrypt(), scramble_encrypt(), and scramble_import().

5.6.1.10 uint32_t hash_fletcher32 (void * *buffer*, size_t *length*)

Computer a 32-bit Fletcher hash for a block of data.

Parameters:

- buffer* a pointer to the data buffer to be checked.
- length* the length, in bytes, of the data to be checked.

Returns:

- a 32-bit number containing the Fletcher hash of the specified data.

Definition at line 16 of file fletcher.c.

Referenced by hashed_bucket().

5.6.1.11 uint32_t hash_murmur32 (void * *buffer*, size_t *length*)

Generate a 32-bit Murmur hash of a block of data.

Parameters:

- buffer* a pointer to the block of data to be hashed.
- length* the length, in bytes, of the block of data to be hashed.

Returns:

- the 32-bit value of the Murmur hash of the specified block of data.

Definition at line 17 of file murmur.c.

References data.

5.6.1.12 uint64_t hash_murmur64 (void * *buffer*, size_t *length*)

Generate a 64-bit Murmur hash of a block of data.

Parameters:

- buffer* a pointer to the block of data to be hashed.
- length* the length, in bytes, of the block of data to be hashed.

Returns:

- the 64-bit value of the Murmur hash of the specified block of data.

Definition at line 72 of file murmur.c.

References data.

5.7 src/core/checksum/crc.c File Reference

```
#include "magma.h"
```

Defines

- #define [A1](#) A
- #define [A\(x\)](#) ((x) & 0xFF)
- #define [B\(x\)](#) (((x) >> 8) & 0xFF)
- #define [C\(x\)](#) (((x) >> 16) & 0xFF)
- #define [D\(x\)](#) ((x) >> 24)
- #define [S8\(x\)](#) ((x) >> 8)
- #define [S32\(x\)](#) ((x) >> 32)
- #define [CRC24_INIT](#) 0xB704CEL
- #define [CRC24_POLY](#) 0x1864CFBL

Functions

- uint32_t [crc24_init](#) (void)
crc.c
- uint32_t [crc24_update](#) (void *buffer, size_t [length](#), uint32_t crc)
- uint32_t [crc24_final](#) (uint32_t crc)
- uint32_t [crc24_checksum](#) (void *buffer, size_t [length](#))
Generate a 24-bit CRC value for the Privacy Enhanced Message format.
- uint32_t [crc32_update](#) (void *buffer, size_t [length](#), uint32_t crc)
Update a 32-bit CRC value with a check of additional data.
- uint32_t [crc32_checksum](#) (void *buffer, size_t [length](#))
Get 32-bit CRC value for a specified block of data.
- uint64_t [crc64_update](#) (void *buffer, size_t [length](#), uint64_t crc)
Update a 64-bit CRC value with a check of additional data.
- uint64_t [crc64_checksum](#) (void *buffer, size_t [length](#))
Get 64-bit CRC value for a specified block of data.

Variables

- const uint32_t [crc24_table](#) [1024]
- const uint32_t [crc32_table](#) [8][256]
- const uint64_t [crc64_table](#) [4][256]

5.7.1 Define Documentation

5.7.1.1 #define A(x) ((x) & 0xFF)

Definition at line 11 of file `crc.c`.

Referenced by `crc32_update()`, `crc64_update()`, `ed25519_publickey()`, and `ed25519_sign_open()`.

5.7.1.2 #define A1 A

Definition at line 10 of file crc.c.

Referenced by crc64_update().

5.7.1.3 #define B(x) (((x) >> 8) & 0xFF)

Definition at line 12 of file crc.c.

Referenced by crc32_update(), and crc64_update().

5.7.1.4 #define C(x) (((x) >> 16) & 0xFF)

Definition at line 13 of file crc.c.

Referenced by crc32_update(), and crc64_update().

5.7.1.5 #define CRC24_INIT 0xB704CEL

Definition at line 23 of file crc.c.

Referenced by crc24_init().

5.7.1.6 #define CRC24_POLY 0x1864CFBL

Definition at line 24 of file crc.c.

Referenced by crc24_checksum(), and crc24_update().

5.7.1.7 #define D(x) ((x) >> 24)

Definition at line 14 of file crc.c.

Referenced by crc32_update(), and crc64_update().

5.7.1.8 #define S32(x) ((x) >> 32)

Definition at line 17 of file crc.c.

Referenced by crc64_update().

5.7.1.9 #define S8(x) ((x) >> 8)

Definition at line 16 of file crc.c.

Referenced by crc32_update(), and crc64_update().

5.7.2 Function Documentation

5.7.2.1 uint32_t crc24_checksum (void * buffer, size_t length)

Generate a 24-bit CRC value for the Privacy Enhanced Message format.

Note:

This function uses the predefined initial value and polynomial defined in RFC 4880.

Parameters:

buffer a pointer to the data to be checked.

length the length, in bytes, of the input buffer.

Returns:

the updated 24-bit CRC value in a 32-bit unsigned integer.

Definition at line 59 of file `crc.c`.

References `crc24_final()`, `crc24_init()`, and `CRC24_POLY`.

Referenced by `prime_pem_unwrap()`, and `prime_pem_wrap()`.

5.7.2.2 `uint32_t crc24_final (uint32_t crc)`

Definition at line 48 of file `crc.c`.

Referenced by `crc24_checksum()`.

5.7.2.3 `uint32_t crc24_init (void)`

[crc.c](#)

Definition at line 26 of file `crc.c`.

References `CRC24_INIT`.

Referenced by `crc24_checksum()`.

5.7.2.4 `uint32_t crc24_update (void * buffer, size_t length, uint32_t crc)`

Definition at line 30 of file `crc.c`.

References `CRC24_POLY`.

5.7.2.5 `uint32_t crc32_checksum (void * buffer, size_t length)`

Get 32-bit CRC value for a specified block of data.

Parameters:

buffer a pointer to the data to be checked.

length the length, in bytes, of the input buffer.

Returns:

the 32-bit CRC value of the specified data.

Definition at line 115 of file `crc.c`.

References `crc32_update()`.

5.7.2.6 uint32_t crc32_update (void * *buffer*, size_t *length*, uint32_t *crc*)

Update a 32-bit CRC value with a check of additional data.

Parameters:

- buffer* a pointer to the data to be checked.
- length* the length, in bytes, of the input buffer.
- crc* the previously computed CRC value, or 0 if this is the initial pass.

Returns:

the updated 32-bit CRC value of the specified data.

Definition at line 83 of file crc.c.

References A, B, C, crc32_table, D, and S8.

Referenced by crc32_checksum().

5.7.2.7 uint64_t crc64_checksum (void * *buffer*, size_t *length*)

Get 64-bit CRC value for a specified block of data.

Parameters:

- buffer* a pointer to the data to be checked.
- length* the length, in bytes, of the input buffer.

Returns:

the 64-bit CRC value of the specified data.

Definition at line 156 of file crc.c.

References crc64_update().

Referenced by smtp_bounce(), and smtp_reply().

5.7.2.8 uint64_t crc64_update (void * *buffer*, size_t *length*, uint64_t *crc*)

Update a 64-bit CRC value with a check of additional data.

Parameters:

- buffer* a pointer to the data to be checked.
- length* the length, in bytes, of the input buffer.
- crc* the previously computed CRC value, or 0 if this is the initial pass.

Returns:

the updated 64-bit CRC value of the specified data.

Definition at line 126 of file crc.c.

References A, A1, B, C, crc64_table, D, S32, and S8.

Referenced by crc64_checksum().

5.7.3 Variable Documentation

5.7.3.1 `const uint32_t crc24_table`

Definition at line 160 of file `crc.c`.

5.7.3.2 `const uint32_t crc32_table`

Definition at line 419 of file `crc.c`.

Referenced by `crc32_update()`.

5.7.3.3 `const uint64_t crc64_table`

Definition at line 943 of file `crc.c`.

Referenced by `crc64_update()`.

5.8 src/core/checksum/fletcher.c File Reference

```
#include "magma.h"
```

Functions

- uint32_t [hash_fletcher32](#) (void *buffer, size_t length)

Computer a 32-bit Fletcher hash for a block of data.

5.8.1 Function Documentation

5.8.1.1 uint32_t hash_fletcher32 (void * *buffer*, size_t *length*)

Computer a 32-bit Fletcher hash for a block of data.

Parameters:

buffer a pointer to the data buffer to be checked.

length the length, in bytes, of the data to be checked.

Returns:

a 32-bit number containing the Fletcher hash of the specified data.

Definition at line 16 of file fletcher.c.

Referenced by hashed_bucket().

5.9 src/core/checksum/murmur.c File Reference

```
#include "magma.h"
```

Functions

- `uint32_t hash_murmur32` (`void *buffer`, `size_t length`)
Generate a 32-bit Murmur hash of a block of data.
- `uint64_t hash_murmur64` (`void *buffer`, `size_t length`)
Generate a 64-bit Murmur hash of a block of data.

5.9.1 Function Documentation

5.9.1.1 `uint32_t hash_murmur32` (`void * buffer`, `size_t length`)

Generate a 32-bit Murmur hash of a block of data.

Parameters:

- buffer* a pointer to the block of data to be hashed.
length the length, in bytes, of the block of data to be hashed.

Returns:

the 32-bit value of the Murmur hash of the specified block of data.

Definition at line 17 of file `murmur.c`.

References `data`.

5.9.1.2 `uint64_t hash_murmur64` (`void * buffer`, `size_t length`)

Generate a 64-bit Murmur hash of a block of data.

Parameters:

- buffer* a pointer to the block of data to be hashed.
length the length, in bytes, of the block of data to be hashed.

Returns:

the 64-bit value of the Murmur hash of the specified block of data.

Definition at line 72 of file `murmur.c`.

References `data`.

5.10 src/core/classify/ascii.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t chr_ascii (uchar_t c)`
Determine whether a specified character is an ASCII character.
- `bool_t chr_printable (uchar_t c)`
Determine whether a specified character is a printable character.
- `bool_t chr_alphanumeric (uchar_t c)`
Determine whether a specified character is alpha-numeric.
- `bool_t chr_lower (uchar_t c)`
Determine whether a specified character is a lowercase character.
- `bool_t chr_upper (uchar_t c)`
Determine whether a specified character is an uppercase character.
- `bool_t chr_numeric (uchar_t c)`
Determine whether a specified character is a numeric character.
- `bool_t chr_punctuation (uchar_t c)`
Determine whether a specified character is a punctuation character.
- `bool_t chr_blank (uchar_t c)`
Determine whether a specified character is a blank character (space or tab).
- `bool_t chr_whitespace (uchar_t c)`
Determine whether a specified character is a whitespace character (blank or).
- `bool_t chr_is_class (uchar_t c, uchar_t *chrs, size_t chrln)`
Determine whether a character belongs to a custom set of values.

5.10.1 Function Documentation

5.10.1.1 `bool_t chr_alphanumeric (uchar_t c)`

Determine whether a specified character is alpha-numeric. [ascii.c](#)

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 43 of file `ascii.c`.

5.10.1.2 `bool_t chr_ascii (uchar_t c)`

Determine whether a specified character is an ASCII character.

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 15 of file `ascii.c`.

5.10.1.3 `bool_t chr_blank (uchar_t c)`

Determine whether a specified character is a blank character (space or tab).

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 113 of file `ascii.c`.

5.10.1.4 `bool_t chr_is_class (uchar_t c, uchar_t * chrs, size_t chrlen)`

Determine whether a character belongs to a custom set of values.

Parameters:

c the character to be verified.

chrs a pointer to a buffer containing the family of valid character values.

chrlen the number of characters in the testing set.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 143 of file `ascii.c`.

5.10.1.5 `bool_t chr_lower (uchar_t c)`

Determine whether a specified character is a lowercase character.

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 57 of file `ascii.c`.

5.10.1.6 `bool_t chr_numeric (uchar_t c)`

Determine whether a specified character is a numeric character.

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 85 of file `ascii.c`.

Referenced by `process_find_pid()`, and `process_kill()`.

5.10.1.7 `bool_t chr_printable (uchar_t c)`

Determine whether a specified character is a printable character.

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 29 of file `ascii.c`.

Referenced by `contact_validate_detail()`, `contact_validate_name()`, and `hex_encode_st_debug()`.

5.10.1.8 `bool_t chr_punctuation (uchar_t c)`

Determine whether a specified character is a punctuation character.

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 99 of file `ascii.c`.

5.10.1.9 `bool_t chr_upper (uchar_t c)`

Determine whether a specified character is an uppercase character.

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 71 of file `ascii.c`.

5.10.1.10 bool_t chr_whitespace (uchar_t *c*)

Determine whether a specified character is a whitespace character (blank or).

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 127 of file ascii.c.

Referenced by auth_sanitise_address(), and imap_command_log_safe().

5.11 src/core/classify/classify.h File Reference

Functions

- [bool_t chr_alphanumeric \(uchar_t c\)](#)
ascii.c
- [bool_t chr_ascii \(uchar_t c\)](#)
Determine whether a specified character is an ASCII character.
- [bool_t chr_blank \(uchar_t c\)](#)
Determine whether a specified character is a blank character (space or tab).
- [bool_t chr_lower \(uchar_t c\)](#)
Determine whether a specified character is a lowercase character.
- [bool_t chr_numeric \(uchar_t c\)](#)
Determine whether a specified character is a numeric character.
- [bool_t chr_printable \(uchar_t c\)](#)
Determine whether a specified character is a printable character.
- [bool_t chr_punctuation \(uchar_t c\)](#)
Determine whether a specified character is a punctuation character.
- [bool_t chr_upper \(uchar_t c\)](#)
Determine whether a specified character is an uppercase character.
- [bool_t chr_whitespace \(uchar_t c\)](#)
Determine whether a specified character is a whitespace character (blank or).
- [bool_t chr_is_class \(uchar_t c, uchar_t *chrs, size_t chrln\)](#)
Determine whether a character belongs to a custom set of values.

5.11.1 Function Documentation

5.11.1.1 bool_t chr_alphanumeric (uchar_t c)

[ascii.c](#) [ascii.c](#)

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 43 of file [ascii.c](#).

5.11.1.2 `bool_t chr_ascii (uchar_t c)`

Determine whether a specified character is an ASCII character.

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 15 of file `ascii.c`.

5.11.1.3 `bool_t chr_blank (uchar_t c)`

Determine whether a specified character is a blank character (space or tab).

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 113 of file `ascii.c`.

5.11.1.4 `bool_t chr_is_class (uchar_t c, uchar_t *chrs, size_t chrlen)`

Determine whether a character belongs to a custom set of values.

Parameters:

c the character to be verified.

chrs a pointer to a buffer containing the family of valid character values.

chrlen the number of characters in the testing set.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 143 of file `ascii.c`.

5.11.1.5 `bool_t chr_lower (uchar_t c)`

Determine whether a specified character is a lowercase character.

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 57 of file `ascii.c`.

5.11.1.6 `bool_t chr_numeric (uchar_t c)`

Determine whether a specified character is a numeric character.

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 85 of file `ascii.c`.

Referenced by `process_find_pid()`, and `process_kill()`.

5.11.1.7 `bool_t chr_printable (uchar_t c)`

Determine whether a specified character is a printable character.

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 29 of file `ascii.c`.

Referenced by `contact_validate_detail()`, `contact_validate_name()`, and `hex_encode_st_debug()`.

5.11.1.8 `bool_t chr_punctuation (uchar_t c)`

Determine whether a specified character is a punctuation character.

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 99 of file `ascii.c`.

5.11.1.9 `bool_t chr_upper (uchar_t c)`

Determine whether a specified character is an uppercase character.

Parameters:

`c` the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 71 of file `ascii.c`.

5.11.1.10 bool_t chr_whitespace (uchar_t *c*)

Determine whether a specified character is a whitespace character (blank or).

Parameters:

c the character to be verified.

Returns:

true if the character passed the test, or false otherwise.

Definition at line 127 of file ascii.c.

Referenced by auth_sanitize_address(), and imap_command_log_safe().

5.12 src/core/compare/compare.h File Reference

Functions

- [int_t st_cmp_ci_ends](#) ([stringer_t](#) *s, [stringer_t](#) *ends)
ends.c
- [int_t st_cmp_cs_ends](#) ([stringer_t](#) *s, [stringer_t](#) *ends)
Perform a case-sensitive check to see if one string ends in another, comparing backwards.
- [int_t mm_cmp_ci_eq](#) (void *a, void *b, [size_t](#) len)
equal.c
- [int_t mm_cmp_cs_eq](#) (void *a, void *b, [size_t](#) len)
Perform a case-sensitive comparison of two memory blocks.
- [int_t st_cmp_ci_eq](#) ([stringer_t](#) *a, [stringer_t](#) *b)
Perform a case-insensitive comparison of two managed strings.
- [int_t st_cmp_cs_eq](#) ([stringer_t](#) *a, [stringer_t](#) *b)
Perform a case-sensitive comparison of two managed strings.
- [bool_t st_search_ci](#) ([stringer_t](#) *haystack, [stringer_t](#) *needle, [size_t](#) *location)
search.c
- [bool_t st_search_cs](#) ([stringer_t](#) *haystack, [stringer_t](#) *needle, [size_t](#) *location)
Search one managed string for an occurrence of another in a case-sensitive manner, and save its location.
- [bool_t st_search_chr](#) ([stringer_t](#) *haystack, [chr_t](#) needle, [size_t](#) *location)
Search for a character inside of a managed string, and save its location.
- [int_t st_cmp_ci_starts](#) ([stringer_t](#) *s, [stringer_t](#) *starts)
starts.c
- [int_t st_cmp_cs_starts](#) ([stringer_t](#) *s, [stringer_t](#) *starts)
Perform a case-sensitive check to see if one string starts with another.

5.12.1 Function Documentation

5.12.1.1 [int_t mm_cmp_ci_eq](#) (void *a, void *b, [size_t](#) len)

[equal.c](#) [equal.c](#)

Parameters:

- a* a pointer to the first buffer in memory to be compared.
- b* a pointer to the second buffer in memory to be compared.
- len* the length, in bytes, of the comparison that is to take place.

Returns:

- 1 if $a < b$, 1 if $b < a$, or 0 if the two memory blocks are equal.

Definition at line 52 of file equal.c.

References `lower_chr()`, and `mm_empty()`.

Referenced by `imap_fetch_body_header()`, `mail_add_forward_headers()`, `mail_add_outbound_headers()`, `mail_discover_encoding()`, `mail_discover_type()`, `mail_header_fetch_all()`, `mail_headers()`, `mail_message()`, `mail_message_cleanup()`, `mail_mime_encoding()`, and `mail_mime_type()`.

5.12.1.2 `int_t mm_cmp_cs_eq (void *a, void *b, size_t len)`

Perform a case-sensitive comparison of two memory blocks.

Parameters:

- a* a pointer to the first buffer in memory to be compared.
- b* a pointer to the second buffer in memory to be compared.
- len* the length, in bytes, of the comparison that is to take place.

Returns:

- 1 if $a < b$, 1 if $b < a$, or 0 if the two memory blocks are equal.

Definition at line 17 of file equal.c.

References `mm_empty()`.

Referenced by `args_parse()`, `mail_get_chunk()`, `mail_mime_child()`, `mail_mime_count()`, and `mail_mime_get_media_type()`.

5.12.1.3 `int_t st_cmp_ci_ends (stringer_t *s, stringer_t *ends)`

[ends.c](#) [ends.c](#)

Note:

The value of the result depends on a backwards character comparison pattern, not a forward one.

Parameters:

- s* the managed string to have its ending characters examined.
- ends* the managed string to be compared against the end of *s*.

Returns:

- 1 if $s < ends$, 1 if $ends < s$ or 0 if *s* ends with the content of *ends*.

Definition at line 60 of file ends.c.

References `lower_chr()`, and `st_empty_out()`.

Referenced by `http_content_load_fonts()`, `http_load_file()`, and `register_captcha_random_font()`.

5.12.1.4 `int_t st_cmp_ci_eq (stringer_t *a, stringer_t *b)`

Perform a case-insensitive comparison of two managed strings.

Parameters:

- a* the first managed string to be compared.
- b* the second managed string to be compared.

Returns:

-1 if $a < b$, 1 if $b < a$, or 0 if the two memory blocks are equal.

Definition at line 126 of file equal.c.

References `lower_chr()`, and `st_empty_out()`.

Referenced by `api_endpoint_register()`, `cache_config()`, `cache_set_value()`, `color_supported()`, `config_key_lookup()`, `config_load_database_settings()`, `config_value_set()`, `contact_edit()`, `contact_find_detail()`, `contact_find_name()`, `dmtplib_compare()`, `http_data_get()`, `http_parse_header()`, `http_response_allow_cross()`, `imap_command_parser()`, `imap_compare()`, `imap_fetch_body()`, `imap_flag_action()`, `imap_folder_compare()`, `imap_folder_create()`, `imap_folder_remove()`, `imap_folder_rename()`, `imap_get_flag()`, `imap_list()`, `imap_lsub()`, `imap_parse_dataitems()`, `imap_search_messages_date_compare()`, `imap_search_messages_inner()`, `imap_status()`, `lib_load()`, `magma_folder_find_full_name()`, `magma_folder_find_name()`, `mail_add_inbound_headers()`, `mail_discover_insertion_point()`, `meta_folders_by_name()`, `molten_compare()`, `pop_compare()`, `portal_endpoint_compare()`, `portal_endpoint_contacts_add()`, `portal_endpoint_folders_list()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_tag()`, `portal_parse_action()`, `portal_parse_context()`, `portal_parse_flags()`, `portal_parse_sections()`, `portal_upload()`, `register_business_step1()`, `relay_config()`, `relay_set_value()`, `servers_config()`, `servers_set_value()`, `servers_validate()`, `smtp_accept_message()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_check_filters()`, `smtp_compare()`, `smtp_data_outbound()`, and `smtp_rcpt_to()`.

5.12.1.5 int_t st_cmp_ci_starts (stringer_t * s, stringer_t * starts)

[starts.c](#) [starts.c](#)

Parameters:

- s* the managed string to have its beginning characters examined.
- starts* the managed string to be compared against the beginning of *s*.

Returns:

-1 if $s < starts$, 1 if $starts < s$ or 0 if *s* begins with *starts*.

Definition at line 54 of file starts.c.

References `lower_chr()`, and `st_empty_out()`.

Referenced by `cache_config()`, `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_file_settings()`, `dmtplib_compare()`, `http_parse_context()`, `http_parse_method()`, `http_parse_origin()`, `http_response()`, `imap_compare()`, `mail_add_forward_headers()`, `mail_count_received()`, `mail_header_fetch_cleaned()`, `mail_mod_subject()`, `molten_compare()`, `pop_compare()`, `portal_endpoint_cookies()`, `register_abuse_check_blocklist()`, `relay_config()`, `servers_config()`, `smtp_compare()`, `smtp_parse_mail_from_path()`, and `virus_check()`.

5.12.1.6 int_t st_cmp_cs_ends (stringer_t * s, stringer_t * ends)

Perform a case-sensitive check to see if one string ends in another, comparing backwards.

Note:

The value of the result depends on a backwards character comparison pattern, not a forward one.

Parameters:

- s* the managed string to have its ending characters examined.
- ends* the managed string to be compared against the end of *s*.

Returns:

-1 if $s < ends$, 1 if $ends < s$ or 0 if *s* ends with the content of *ends*.

Definition at line 17 of file ends.c.

References st_empty_out().

Referenced by folder_count(), http_content_load_directory(), http_content_load_fonts(), http_content_start(), http_load_file(), smtp_client_send_data(), and stats_sum_errors().

5.12.1.7 int_t st_cmp_cs_eq (stringer_t * a, stringer_t * b)

Perform a case-sensitive comparison of two managed strings.

Parameters:

- a* the first managed string to be compared.
- b* the second managed string to be compared.

Returns:

- 1 if $a < b$, 1 if $b < a$, or 0 if the two memory blocks are equal.

Definition at line 85 of file equal.c.

References st_empty_out().

Referenced by args_parse(), auth_data_fetch(), auth_login(), auth_response(), auth_sanitise_username(), cmp_mt_mt(), config_free(), config_output_value(), config_output_value_generic(), config_value_set(), contact_business(), contact_business_add_error(), contact_print_form(), contact_print_message(), contact_process(), encrypted_chunk_get(), folder_count(), get_header_opt(), http_content_load_directory(), http_load_file(), http_response(), ident_mt_mt(), imap_fetch(), imap_fetch_bodystructure(), magma_folder_find_full_name(), magma_folder_find_name(), mail_add_inbound_headers(), mail_add_outbound_headers(), mail_get_chunk(), meta_folders_by_name(), meta_get(), portal_endpoint_contacts_add(), prime_pem_unwrap(), process_find_pid(), register_business_step2(), sanity_check(), servers_output_settings(), sess_get(), slots_key(), smtp_accept_message(), smtp_bounce(), smtp_get_action(), smtp_parse_mail_from_path(), smtp_update_receive_stats(), spool_check_file(), STACK_OF(), stats_get_name_pos(), stats_sum_errors(), teacher_process(), tls_cipher(), tls_version(), and user_config_edit().

5.12.1.8 int_t st_cmp_cs_starts (stringer_t * s, stringer_t * starts)

Perform a case-sensitive check to see if one string starts with another.

Parameters:

- s* the managed string to have its beginning characters examined.
- starts* the managed string to be compared against the beginning of *s*.

Returns:

- 1 if $s < starts$, 1 if $starts < s$ or 0 if *s* begins with *starts*.

Definition at line 16 of file starts.c.

References st_empty_out().

Referenced by http_response(), imap_parse_dataitems(), mail_add_forward_headers(), process_kill(), st_replace(), stats_sum_errors(), and teacher_process().

5.12.1.9 bool_t st_search_chr (stringer_t * haystack, chr_t needle, size_t * location)

Search for a character inside of a managed string, and save its location.

Parameters:

- haystack* the managed string to be searched.

needle the character to be found in the string.

location if not NULL, a pointer to store the index of needle if found, or 0 on no match.

Returns:

true if the specified character was found or false otherwise.

Definition at line 117 of file search.c.

References log_pedantic, and st_empty_out().

Referenced by auth_data_fetch(), and register_data_insert_user().

5.12.1.10 bool_t st_search_ci (stringer_t * haystack, stringer_t * needle, size_t * location)

search.c search.c

Parameters:

haystack the managed string to be searched.

needle the managed string to be found.

location if not NULL, a pointer to store the index of needle if found, or 0 on no match.

Returns:

true if the string is found or false otherwise.

Definition at line 68 of file search.c.

References log_pedantic, lower_chr(), and st_empty_out().

Referenced by imap_command_log_safe(), imap_search_messages_body(), imap_search_messages_header(), imap_search_messages_text(), mail_get_boundary(), mail_mime_boundary(), mail_mime_generate_boundary(), and pattern_check().

5.12.1.11 bool_t st_search_cs (stringer_t * haystack, stringer_t * needle, size_t * location)

Search one managed string for an occurrence of another in a case-sensitive manner, and save its location.

Parameters:

haystack the managed string to be searched.

needle the managed string to be found.

location if not NULL, a pointer to store the index of needle if found, or 0 on no match.

Returns:

true if the string is found or false otherwise.

Definition at line 17 of file search.c.

References log_pedantic, and st_empty_out().

Referenced by auth_sanitize_username(), http_parse_header(), http_parse_pairs(), mail_get_chunk(), str_tok_get_bl(), and str_tok_get_count_bl().

5.13 src/core/compare/ends.c File Reference

```
#include "magma.h"
```

Functions

- [int_t st_cmp_cs_ends](#) ([stringer_t](#) *s, [stringer_t](#) *ends)
Perform a case-sensitive check to see if one string ends in another, comparing backwards.
- [int_t st_cmp_ci_ends](#) ([stringer_t](#) *s, [stringer_t](#) *ends)
Perform a case-insensitive check to see if one string ends in another, comparing backwards.

5.13.1 Function Documentation

5.13.1.1 int_t st_cmp_ci_ends (stringer_t * s, stringer_t * ends)

Perform a case-insensitive check to see if one string ends in another, comparing backwards. [ends.c](#)

Note:

The value of the result depends on a backwards character comparison pattern, not a forward one.

Parameters:

- s* the managed string to have its ending characters examined.
- ends* the managed string to be compared against the end of *s*.

Returns:

-1 if *s* < *ends*, 1 if *ends* < *s* or 0 if *s* ends with the content of *ends*.

Definition at line 60 of file [ends.c](#).

References [lower_chr\(\)](#), and [st_empty_out\(\)](#).

Referenced by [http_content_load_fonts\(\)](#), [http_load_file\(\)](#), and [register_captcha_random_font\(\)](#).

5.13.1.2 int_t st_cmp_cs_ends (stringer_t * s, stringer_t * ends)

Perform a case-sensitive check to see if one string ends in another, comparing backwards.

Note:

The value of the result depends on a backwards character comparison pattern, not a forward one.

Parameters:

- s* the managed string to have its ending characters examined.
- ends* the managed string to be compared against the end of *s*.

Returns:

-1 if *s* < *ends*, 1 if *ends* < *s* or 0 if *s* ends with the content of *ends*.

Definition at line 17 of file [ends.c](#).

References [st_empty_out\(\)](#).

Referenced by [folder_count\(\)](#), [http_content_load_directory\(\)](#), [http_content_load_fonts\(\)](#), [http_content_start\(\)](#), [http_load_file\(\)](#), [smtp_client_send_data\(\)](#), and [stats_sum_errors\(\)](#).

5.14 src/core/compare/equal.c File Reference

```
#include "magma.h"
```

Functions

- [int_t mm_cmp_cs_eq](#) (void *a, void *b, size_t len)
Perform a case-sensitive comparison of two memory blocks.
- [int_t mm_cmp_ci_eq](#) (void *a, void *b, size_t len)
Perform a case-insensitive comparison of two memory blocks.
- [int_t st_cmp_cs_eq](#) (stringer_t *a, stringer_t *b)
Perform a case-sensitive comparison of two managed strings.
- [int_t st_cmp_ci_eq](#) (stringer_t *a, stringer_t *b)
Perform a case-insensitive comparison of two managed strings.

5.14.1 Function Documentation

5.14.1.1 int_t mm_cmp_ci_eq (void *a, void *b, size_t len)

Perform a case-insensitive comparison of two memory blocks. [equal.c](#)

Parameters:

- a** a pointer to the first buffer in memory to be compared.
- b** a pointer to the second buffer in memory to be compared.
- len** the length, in bytes, of the comparison that is to take place.

Returns:

- 1 if a < b, 1 if b < a, or 0 if the two memory blocks are equal.

Definition at line 52 of file equal.c.

References [lower_chr\(\)](#), and [mm_empty\(\)](#).

Referenced by [imap_fetch_body_header\(\)](#), [mail_add_forward_headers\(\)](#), [mail_add_outbound_headers\(\)](#), [mail_discover_encoding\(\)](#), [mail_discover_type\(\)](#), [mail_header_fetch_all\(\)](#), [mail_headers\(\)](#), [mail_message\(\)](#), [mail_message_cleanup\(\)](#), [mail_mime_encoding\(\)](#), and [mail_mime_type\(\)](#).

5.14.1.2 int_t mm_cmp_cs_eq (void *a, void *b, size_t len)

Perform a case-sensitive comparison of two memory blocks.

Parameters:

- a** a pointer to the first buffer in memory to be compared.
- b** a pointer to the second buffer in memory to be compared.
- len** the length, in bytes, of the comparison that is to take place.

Returns:

- 1 if a < b, 1 if b < a, or 0 if the two memory blocks are equal.

Definition at line 17 of file equal.c.

References mm_empty().

Referenced by args_parse(), mail_get_chunk(), mail_mime_child(), mail_mime_count(), and mail_mime_get_media_type().

5.14.1.3 int_t st_cmp_ci_eq (stringer_t * a, stringer_t * b)

Perform a case-insensitive comparison of two managed strings.

Parameters:

- a* the first managed string to be compared.
- b* the second managed string to be compared.

Returns:

- 1 if $a < b$, 1 if $b < a$, or 0 if the two memory blocks are equal.

Definition at line 126 of file equal.c.

References lower_chr(), and st_empty_out().

Referenced by api_endpoint_register(), cache_config(), cache_set_value(), color_supported(), config_key_lookup(), config_load_database_settings(), config_value_set(), contact_edit(), contact_find_detail(), contact_find_name(), dmtip_compare(), http_data_get(), http_parse_header(), http_response_allow_cross(), imap_command_parser(), imap_compare(), imap_fetch_body(), imap_flag_action(), imap_folder_compare(), imap_folder_create(), imap_folder_remove(), imap_folder_rename(), imap_get_flag(), imap_list(), imap_lsub(), imap_parse_dataitems(), imap_search_messages_date_compare(), imap_search_messages_inner(), imap_status(), lib_load(), magma_folder_find_full_name(), magma_folder_find_name(), mail_add_inbound_headers(), mail_discover_insertion_point(), meta_folders_by_name(), molten_compare(), pop_compare(), portal_endpoint_compare(), portal_endpoint_contacts_add(), portal_endpoint_folders_list(), portal_endpoint_messages_flag(), portal_endpoint_messages_tag(), portal_parse_action(), portal_parse_context(), portal_parse_flags(), portal_parse_sections(), portal_upload(), register_business_step1(), relay_config(), relay_set_value(), servers_config(), servers_set_value(), servers_validate(), smtp_accept_message(), smtp_auth_login(), smtp_auth_plain(), smtp_check_filters(), smtp_compare(), smtp_data_outbound(), and smtp_rcpt_to().

5.14.1.4 int_t st_cmp_cs_eq (stringer_t * a, stringer_t * b)

Perform a case-sensitive comparison of two managed strings.

Parameters:

- a* the first managed string to be compared.
- b* the second managed string to be compared.

Returns:

- 1 if $a < b$, 1 if $b < a$, or 0 if the two memory blocks are equal.

Definition at line 85 of file equal.c.

References st_empty_out().

Referenced by args_parse(), auth_data_fetch(), auth_login(), auth_response(), auth_sanitise_username(), cmp_mt_mt(), config_free(), config_output_value(), config_output_value_generic(), config_value_set(), contact_business(), contact_business_add_error(), contact_print_form(), contact_print_message(), contact_process(), encrypted_chunk_get(), folder_count(), get_header_opt(), http_content_load_directory(), http_load_file(), http_response(), ident_mt_mt(), imap_fetch(), imap_fetch_bodystructure(), magma_folder_find_full_name(), magma_folder_find_name(), mail_add_inbound_headers(), mail_add_outbound_headers(), mail_get_chunk(), meta_folders_by_name(), meta_get(), portal_endpoint_contacts_add(), prime_pem_unwrap(), process_find_pid(), register_business_step2(), sanity_check(), servers_output_settings(), sess_get(), slots_key(), smtp_accept_message(), smtp_bounce(), smtp_get_action(), smtp_parse_mail_from_path(), smtp_update_receive_stats(), spool_check_file(), STACK_OF(), stats_get_name_pos(), stats_sum_errors(), teacher_process(), tls_cipher(), tls_version(), and user_config_edit().

5.15 src/core/compare/search.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t st_search_cs (stringer_t *haystack, stringer_t *needle, size_t *location)`
Search one managed string for an occurrence of another in a case-sensitive manner, and save its location.
- `bool_t st_search_ci (stringer_t *haystack, stringer_t *needle, size_t *location)`
Search one managed string for an occurrence of another in a case-insensitive manner, and save its location.
- `bool_t st_search_chr (stringer_t *haystack, chr_t needle, size_t *location)`
Search for a character inside of a managed string, and save its location.

5.15.1 Function Documentation

5.15.1.1 `bool_t st_search_chr (stringer_t *haystack, chr_t needle, size_t *location)`

Search for a character inside of a managed string, and save its location.

Parameters:

haystack the managed string to be searched.
needle the character to be found in the string.
location if not NULL, a pointer to store the index of needle if found, or 0 on no match.

Returns:

true if the specified character was found or false otherwise.

Definition at line 117 of file search.c.

References `log_pedantic`, and `st_empty_out()`.

Referenced by `auth_data_fetch()`, and `register_data_insert_user()`.

5.15.1.2 `bool_t st_search_ci (stringer_t *haystack, stringer_t *needle, size_t *location)`

Search one managed string for an occurrence of another in a case-insensitive manner, and save its location. search.c

Parameters:

haystack the managed string to be searched.
needle the managed string to be found.
location if not NULL, a pointer to store the index of needle if found, or 0 on no match.

Returns:

true if the string is found or false otherwise.

Definition at line 68 of file search.c.

References `log_pedantic`, `lower_chr()`, and `st_empty_out()`.

Referenced by `imap_command_log_safe()`, `imap_search_messages_body()`, `imap_search_messages_header()`, `imap_search_messages_text()`, `mail_get_boundary()`, `mail_mime_boundary()`, `mail_mime_generate_boundary()`, and `pattern_check()`.

5.15.1.3 `bool_t st_search_cs (stringer_t * haystack, stringer_t * needle, size_t * location)`

Search one managed string for an occurrence of another in a case-sensitive manner, and save its location.

Parameters:

haystack the managed string to be searched.

needle the managed string to be found.

location if not NULL, a pointer to store the index of needle if found, or 0 on no match.

Returns:

true if the string is found or false otherwise.

Definition at line 17 of file search.c.

References `log_pedantic`, and `st_empty_out()`.

Referenced by `auth_sanitize_username()`, `http_parse_header()`, `http_parse_pairs()`, `mail_get_chunk()`, `str_tok_get_bl()`, and `str_tok_get_count_bl()`.

5.16 src/servers/imap/search.c File Reference

```
#include "magma.h"
```

Functions

- `int_t imap_search_messages_date_compare (stringer_t *one, stringer_t *two)`
- `int_t imap_search_messages_date (connection_t *con, meta_user_t *user, mail_message_t **data, stringer_t **header, meta_message_t *active, stringer_t *date, int_t internal, int_t expected)`
- `int_t imap_search_messages_header (connection_t *con, meta_user_t *user, mail_message_t **data, stringer_t **header, meta_message_t *active, stringer_t *field, stringer_t *value)`
- `int_t imap_search_messages_body (connection_t *con, meta_user_t *user, mail_message_t **data, meta_message_t *active, stringer_t *value)`
- `int_t imap_search_messages_text (connection_t *con, meta_user_t *user, mail_message_t **data, meta_message_t *active, stringer_t *value)`
- `int_t imap_search_messages_size (meta_message_t *active, stringer_t *value, int_t expected)`
- `int_t imap_search_flag (uint32_t status, uint32_t flag, int_t has)`

search.c

- `int_t imap_search_messages_range (meta_message_t *active, stringer_t *range, int_t uid)`
- `int_t imap_search_messages_inner (connection_t *con, meta_user_t *user, mail_message_t **message, stringer_t **header, meta_message_t *current, imap_arguments_t *array, unsigned recursion)`
- `inx_t * imap_search_messages (connection_t *con)`

Variables

- `chr_t * MONTH_LOOKUP [] = { "JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC" }`

5.16.1 Function Documentation

5.16.1.1 `int_t imap_search_flag (uint32_t status, uint32_t flag, int_t has)`

search.c

Definition at line 266 of file *search.c*.

Referenced by `imap_search_messages_inner()`.

5.16.1.2 `inx_t* imap_search_messages (connection_t * con)`

LOW: Is a read lock necessary now that were using index reference counters and thread safe iteration cursors?

Definition at line 563 of file *search.c*.

References `count`, `imap_search_messages_inner()`, `inx_alloc()`, `inx_append()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_key_active()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_find()`, `inx_t`, `M_INX_LINKED`, `M_TYPE_UINT64`, `mail_destroy()`, `mail_destroy_header()`, `meta_message_dupe()`, `meta_message_free()`, `meta_message_t`, `meta_user_rlock()`, `meta_user_unlock()`, `status`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_search()`.

5.16.1.3 `int_t imap_search_messages_body (connection_t * con, meta_user_t * user, mail_message_t ** data, meta_message_t * active, stringer_t * value)`

Definition at line 197 of file *search.c*.

References mail_load_message(), mail_mime_update(), st_free(), and st_search_ci().

Referenced by imap_search_messages_inner().

5.16.1.4 int_t imap_search_messages_date (connection_t * con, meta_user_t * user, mail_message_t ** data, stringer_t ** header, meta_message_t * active, stringer_t * date, int_t internal, int_t expected)

Definition at line 82 of file search.c.

References imap_search_messages_date_compare(), mail_header_fetch_cleaned(), mail_load_header(), ns_length_get(), pl_empty(), pl_init(), pl_null(), PLACER, placer_t, st_char_get(), st_free(), st_import(), st_length_get(), st_merge, tok_get_count_st(), and tok_get_st().

Referenced by imap_search_messages_inner().

5.16.1.5 int_t imap_search_messages_date_compare (stringer_t * one, stringer_t * two)

Definition at line 12 of file search.c.

References MONTH_LOOKUP, PLACER, placer_t, st_cmp_ci_eq(), tok_get_count_st(), tok_get_st(), and uint32_conv_st().

Referenced by imap_search_messages_date().

5.16.1.6 int_t imap_search_messages_header (connection_t * con, meta_user_t * user, mail_message_t ** data, stringer_t ** header, meta_message_t * active, stringer_t * field, stringer_t * value)

Definition at line 161 of file search.c.

References mail_header_fetch_all(), mail_load_header(), pl_empty(), pl_init(), pl_null(), placer_t, st_char_get(), st_free(), st_length_get(), and st_search_ci().

Referenced by imap_search_messages_inner().

5.16.1.7 int_t imap_search_messages_inner (connection_t * con, meta_user_t * user, mail_message_t ** message, stringer_t ** header, meta_message_t * current, imap_arguments_t * array, unsigned recursion)

Definition at line 347 of file search.c.

References ar_length_get(), IMAP_ARGUMENT_TYPE_ARRAY, imap_get_ar_ar(), imap_get_st_ar(), imap_get_type_ar(), imap_search_flag(), imap_search_messages_body(), imap_search_messages_date(), imap_search_messages_header(), imap_search_messages_inner(), imap_search_messages_range(), imap_search_messages_size(), imap_search_messages_text(), IMAP_SEARCH_RECURSION_LIMIT, imap_valid_sequence(), log_pedantic, MAIL_STATUS_ANSWERED, MAIL_STATUS_DELETED, MAIL_STATUS_DRAFT, MAIL_STATUS_FLAGGED, MAIL_STATUS_RECENT, MAIL_STATUS_SEEN, number, PLACER, and st_cmp_ci_eq().

Referenced by imap_search_messages(), and imap_search_messages_inner().

5.16.1.8 int_t imap_search_messages_range (meta_message_t * active, stringer_t * range, int_t uid)

Definition at line 276 of file search.c.

References number, pl_char_get(), pl_empty(), pl_null(), placer_t, tok_get_count_st(), tok_get_st(), and uint64_conv_st().

Referenced by imap_search_messages_inner().

5.16.1.9 int_t imap_search_messages_size (meta_message_t * active, stringer_t * value, int_t expected)

Definition at line 245 of file search.c.

References uint64_conv_st().

Referenced by imap_search_messages_inner().

5.16.1.10 `int_t imap_search_messages_text (connection_t * con, meta_user_t * user, mail_message_t ** data, meta_message_t * active, stringer_t * value)`

Definition at line 221 of file search.c.

References mail_load_message(), mail_mime_update(), st_free(), and st_search_ci().

Referenced by imap_search_messages_inner().

5.16.2 Variable Documentation

5.16.2.1 `chr_t* MONTH_LOOKUP[] = { "JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC" }`

Definition at line 10 of file search.c.

Referenced by imap_search_messages_date_compare().

5.17 src/core/compare/starts.c File Reference

```
#include "magma.h"
```

Functions

- `int_t st_cmp_cs_starts (stringer_t *s, stringer_t *starts)`
Perform a case-sensitive check to see if one string starts with another.
- `int_t st_cmp_ci_starts (stringer_t *s, stringer_t *starts)`
Perform a case-insensitive check to see if one string starts with another.

5.17.1 Function Documentation

5.17.1.1 `int_t st_cmp_ci_starts (stringer_t *s, stringer_t *starts)`

Perform a case-insensitive check to see if one string starts with another. [starts.c](#)

Parameters:

- `s` the managed string to have its beginning characters examined.
- `starts` the managed string to be compared against the beginning of `s`.

Returns:

- 1 if `s < starts`, 1 if `starts < s` or 0 if `s` begins with `starts`.

Definition at line 54 of file `starts.c`.

References `lower_chr()`, and `st_empty_out()`.

Referenced by `cache_config()`, `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_file_settings()`, `dmtmp_compare()`, `http_parse_context()`, `http_parse_method()`, `http_parse_origin()`, `http_response()`, `imap_compare()`, `mail_add_forward_headers()`, `mail_count_received()`, `mail_header_fetch_cleaned()`, `mail_mod_subject()`, `molten_compare()`, `pop_compare()`, `portal_endpoint_cookies()`, `register_abuse_check_blocklist()`, `relay_config()`, `servers_config()`, `smtp_compare()`, `smtp_parse_mail_from_path()`, and `virus_check()`.

5.17.1.2 `int_t st_cmp_cs_starts (stringer_t *s, stringer_t *starts)`

Perform a case-sensitive check to see if one string starts with another.

Parameters:

- `s` the managed string to have its beginning characters examined.
- `starts` the managed string to be compared against the beginning of `s`.

Returns:

- 1 if `s < starts`, 1 if `starts < s` or 0 if `s` begins with `starts`.

Definition at line 16 of file `starts.c`.

References `st_empty_out()`.

Referenced by `http_response()`, `imap_parse_dataitems()`, `mail_add_forward_headers()`, `process_kill()`, `st_replace()`, `stats_sum_errors()`, and `teacher_process()`.

5.18 src/core/core.h File Reference

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stddef.h>
#include <unistd.h>
#include <signal.h>
#include <stdint.h>
#include <stdarg.h>
#include <stdbool.h>
#include <pthread.h>
#include <fcntl.h>
#include <math.h>
#include <semaphore.h>
#include <dirent.h>
#include <limits.h>
#include <ftw.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <sys/utsname.h>
#include <sys/resource.h>
#include "memory/memory.h"
#include "strings/strings.h"
#include "classify/classify.h"
#include "encodings/encodings.h"
#include "indexes/indexes.h"
#include "compare/compare.h"
#include "thread/thread.h"
#include "buckets/buckets.h"
#include "parsers/parsers.h"
#include "checksum/checksum.h"
#include "host/host.h"
```

Defines

- `#define __bool_t_defined`

- #define `INT24_MIN` (-8388607)
- #define `INT24_MAX` (8388607)
- #define `UINT24_MIN` (0)
- #define `UINT24_MAX` (16777215)
- #define `SIGUNUSED` 31
- #define `log_pedantic(...)` do { mutex_lock(&`log_mutex`); if (`log_enabled`) printf(__VA_ARGS__); mutex_unlock(&`log_mutex`); } while (0)
- #define `log_check(expr)` do { } while (0)
- #define `log_info(...)` do { mutex_lock(&`log_mutex`); if (`log_enabled`) printf(__VA_ARGS__); mutex_unlock(&`log_mutex`); } while (0)
- #define `log_error(...)` do { mutex_lock(&`log_mutex`); if (`log_enabled`) printf(__VA_ARGS__); mutex_unlock(&`log_mutex`); } while (0)
- #define `log_critical(...)` do { mutex_lock(&`log_mutex`); if (`log_enabled`) printf(__VA_ARGS__); mutex_unlock(&`log_mutex`); } while (0)
- #define `log_options(options,...)` do { mutex_lock(&`log_mutex`); if (`log_enabled`) printf(__VA_ARGS__); mutex_unlock(&`log_mutex`); } while (0)

Typedefs

- typedef char `bool_t`
- typedef char `chr_t`
- typedef unsigned char `uchr_t`
- typedef unsigned char `byte_t`
- typedef int32_t `int_t`
- typedef uint32_t `uint_t`
- typedef `__int24_t` `int24_t`
- typedef `__uint24_t` `uint24_t`

Enumerations

- enum `M_TYPE` {
`M_TYPE_EMPTY` = 0, `M_TYPE_MULTI` = 1, `M_TYPE_ENUM`, `M_TYPE_BOOLEAN`,
`M_TYPE_BLOCK`, `M_TYPE_NULLER`, `M_TYPE_PLACER`, `M_TYPE_STRINGER`,
`M_TYPE_INT8`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`,
`M_TYPE_UINT8`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`,
`M_TYPE_FLOAT`, `M_TYPE_DOUBLE` }
 • enum { `EMPTY` = 0 }

Functions

- struct `__attribute__` ((packed))
- char * `type` (`M_TYPE` type)

Variables

- `__int24_t`
- `__uint24_t`
- `bool_t` `log_enabled`
- `pthread_mutex_t` `log_mutex`

5.18.1 Define Documentation

5.18.1.1 `#define __bool_t_defined`

Definition at line 48 of file core.h.

5.18.1.2 `#define INT24_MAX (8388607)`

Definition at line 108 of file core.h.

5.18.1.3 `#define INT24_MIN (-8388607)`

Definition at line 104 of file core.h.

5.18.1.4 `#define log_check(expr) do { while (0)`

Definition at line 190 of file core.h.

Referenced by `contact_delete()`, `contact_detail_delete()`, `contact_detail_upsert()`, `contact_update()`, `contact_update_stamp()`, `hashed_bucket()`, `pool_get_status()`, `pool_set_status()`, `protocol_enqueue()`, `protocol_secure()`, `protocol_type()`, `st_swap()`, `tank_load()`, `user_config_delete()`, `user_config_upsert()`, `virus_engine_destroy()`, and `warehouse_fetch_domains()`.

5.18.1.5 `#define log_critical(...) do { mutex_lock(&log_mutex); if (log_enabled) printf(__VA_ARGS__); mutex_unlock(&log_mutex); } while (0)`

Definition at line 193 of file core.h.

Referenced by `args_parse()`, `cache_alloc()`, `cache_config()`, `cache_set_value()`, `cache_start()`, `cache_validate()`, `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_file_settings()`, `config_validate_settings()`, `config_value_set()`, `dkim_start()`, `lib_load()`, `lib_load_lzo()`, `log_rotate()`, `log_start()`, `net_init()`, `net_trigger()`, `obj_cache_start()`, `prime_start()`, `process_start()`, `process_stop()`, `relay_alloc()`, `relay_config()`, `relay_set_value()`, `relay_validate()`, `requeue()`, `servers_alloc()`, `servers_config()`, `servers_set_value()`, `servers_validate()`, `signal_shutdown()`, `spool_check()`, `spool_mktemp()`, `spool_start()`, `sql_open()`, `sql_start()`, `ssl_start()`, `STACK_OF()`, `stats_init()`, `stmt_bind_param()`, `stmt_close()`, `stmt_open()`, `stmt_prepare()`, `stmt_rebuild()`, `stmt_reset()`, `stmt_start()`, `system_fork_daemon()`, `system_init_impersonation()`, `tank_delete()`, `tank_open()`, `tank_start()`, `tls_server_create()`, and `virus_start()`.

5.18.1.6 `#define log_error(...) do { mutex_lock(&log_mutex); if (log_enabled) printf(__VA_ARGS__); mutex_unlock(&log_mutex); } while (0)`

Definition at line 192 of file core.h.

Referenced by `aes_cipher_key()`, `aes_tag_shard()`, `aes_vector_shard()`, `ar_alloc()`, `auth_challenge()`, `auth_data_fetch()`, `auth_data_update_legacy()`, `auth_legacy()`, `auth_login()`, `base64_encode()`, `base64_encode_wrap()`, `chunk_header_write()`, `config_fetch_host_number()`, `contact_business()`, `deprecated_ecies_group()`, `deprecated_ecies_start()`, `dequeue()`, `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_params_generate()`, `dspam_check()`, `ecies_group()`, `ecies_start()`, `hex_decode_opts()`, `hex_encode_opts()`, `hex_encode_st_debug()`, `imap_build_array()`, `imap_duplicate_messages()`, `imap_fetch_response_add()`, `imap_folder_remove()`, `imap_folder_rename()`, `imap_narrow_messages()`, `imap_parse_dataitems()`, `imap_parse_nstring()`, `imap_update_flags()`, `json_api_dispatch()`, `mail_copy_message()`, `mail_create_directory()`, `mail_db_delete_message()`, `mail_get_chunk()`, `mail_mime_boundary()`, `mail_mime_generate_boundary()`, `mail_mime_get_smtp_envelope()`, `mail_move_message()`, `mail_store_message()`, `mail_store_message_data()`, `main()`, `meta_data_fetch_alerts()`, `meta_data_fetch_all_tags()`, `meta_data_fetch_folder_messages()`, `meta_data_fetch_folders()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_messages()`, `meta_messages_copier()`, `net_set_blocking()`, `pop_pass_parse()`, `portal_config_collection()`, `portal_contact_details()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tags()`, `portal_parse_json_str_array()`, `portal_upload()`, `prime_field_write()`, `prime_header_write()`, `prime_start()`, `queue_init()`, `sess_get()`, `signal_refresh()`, `smtp_accept_message()`, `smtp_add_bypass_entry()`, `smtp_fetch_authorization()`, `smtp_fetch_inbound()`, `smtp_get_action()`, `smtp_store_message()`, `spool_check_file()`, `spool_cleanup()`,

sql_ping(), sql_thread_start(), ssl_ecdh_exchange_callback(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_encrypt(), stacie_realm_cipher(), stacie_realm_key(), stacie_realm_tag(), stacie_realm_vector(), stacker_push(), tank_close(), tank_delete(), tank_load(), tank_store(), virus_check(), virus_engine_create(), virus_engine_refresh(), and virus_sigs_total().

5.18.1.7 #define log_info(...) do { mutex_lock(&log_mutex); if (log_enabled) printf(__VA_ARGS__); mutex_unlock(&log_mutex); } while (0)

Definition at line 191 of file core.h.

Referenced by api_endpoint_auth(), args_parse(), cache_add(), cache_append(), cache_delete(), cache_flush(), cache_get(), cache_get_u64(), cache_output_help(), cache_output_settings(), cache_set(), cache_set_u64(), compress_bzip(), compress_lzo(), compress_zlib(), con_init_network_buffer(), config_load_defaults(), config_output_help(), config_output_settings(), config_output_value(), config_output_value_generic(), contact_details_fetch(), contacts_fetch(), decompress_block_lzo(), decompress_bzip(), decompress_lzo(), decompress_zlib(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_ecies_key_alloc(), deprecated_ecies_key_create(), deprecated_ecies_key_private(), deprecated_ecies_key_private_bin(), deprecated_ecies_key_public(), deprecated_ecies_key_public_bin(), deprecated_ecies_key_public_hex(), deprecated_scramble_decrypt(), display_usage(), dspam_check(), dspam_train(), ecies_decrypt(), ecies_encrypt(), ecies_key_alloc(), ecies_key_create(), ecies_key_private(), ecies_key_private_bin(), ecies_key_public(), ecies_key_public_bin(), ecies_key_public_hex(), ed25519_private_set(), ed25519_public_set(), ed25519_sign(), ed25519_verify(), file_load(), file_read(), folder_count(), http_content_load_directory(), http_content_load_fonts(), http_load_file(), http_parse_header(), http_parse_method(), imap_command_log_safe(), imap_id(), imap_login(), linked_append(), linked_insert(), magma_folder_fetch(), mail_load_message(), meta_crypto_keys_create(), meta_data_fetch_alerts(), meta_data_fetch_mailbox_aliases(), net_trigger(), ns_dupe(), nvp_alloc(), pool_alloc(), pop_pass(), portal_endpoint_auth(), process_stop(), queue_signal(), register_print_captcha(), relay_output_help(), relay_output_settings(), res_bind_create(), res_field_count(), res_field_generic(), res_field_length(), res_field_string(), res_row_count(), res_row_get(), res_row_set(), res_row_store(), res_stmt_store(), res_table_alloc(), res_table_free(), scramble_decrypt(), secp256k1_alloc(), secp256k1_compute_kek(), secp256k1_generate(), secp256k1_private_set(), secp256k1_public_set(), servers_output_help(), servers_output_settings(), signal_refresh(), signal_start(), signal_status(), signal_thread_start(), smtp_auth_login(), smtp_auth_plain(), spool_check(), sql_query(), stats_get_name_pos(), stmt_exec(), stmt_exec_affected(), stmt_exec_affected_conn(), stmt_exec_conn(), stmt_get_result(), stmt_get_result_conn(), stmt_insert(), stmt_insert_conn(), system_change_root_directory(), system_fork_daemon(), system_init_core_dumps(), system_init_impersonation(), system_init_resource_limits(), system_init_umask(), system_ulimit_cur(), system_ulimit_max(), tran_commit(), tran_rollback(), tran_start(), tree_insert(), user_config_fetch(), virus_engine_refresh(), and warehouse_fetch_domains().

5.18.1.8 #define log_options(options, ...) do { mutex_lock(&log_mutex); if (log_enabled) printf(__VA_ARGS__); mutex_unlock(&log_mutex); } while (0)

Definition at line 194 of file core.h.

Referenced by cache_output_help(), cmp_mt_mt(), config_output_help(), http_print_500(), http_print_500_log(), ident_mt_mt(), inx_alloc(), lib_load(), mm_sec_alloc(), mm_sec_free(), mt_get_char(), mt_get_length(), mt_get_number(), mt_get_type(), mt_is_empty(), mt_is_number(), mt_set_type(), mutex_lock(), mutex_unlock(), nvp_parse(), portal_config_collection(), portal_endpoint_error(), process_start(), process_stop(), relay_output_help(), servers_output_help(), and signal_segfault().

5.18.1.9 #define log_pedantic(...) do { mutex_lock(&log_mutex); if (log_enabled) printf(__VA_ARGS__); mutex_unlock(&log_mutex); } while (0)

Definition at line 189 of file core.h.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), alert_alloc(), alias_alloc(), api_endpoint_auth(), api_endpoint_change_password(), api_endpoint_delete_user(), api_endpoint_register(), api_response(), ar_alloc(), ar_append(), ar_avail_get(), ar_field_ar(), ar_field_ns(), ar_field_pl(), ar_field_ptr(), ar_field_st(), ar_field_type(), ar_free(), ar_length_get(), ar_length_set(), auth_alloc(), auth_challenge(), auth_data_fetch(), auth_data_update_legacy(), auth_data_update_lock(), auth_free(), auth_login(), auth_response(), auth_stacie(), auth_stacie_alloc(), auth_stacie_free(), base64_decode(), base64_decode_mod(), base64_decode_opts(), base64_encode(), base64_encode_mod(), base64_encode_opts(), base64_encode_wrap(), base64_encoded_length_wrap(), bracket_extract_pl(), cache_decrement(), cache_free(), cache_increment(), cache_output_settings(), cache_validate(), chunk_buffer_read(), chunk_buffer_size(), chunk_header_read(), chunk_header_size(), chunk_header_type(), chunk_header_write(), cipher_block_length(), cipher_id(), cipher_key_length(), cipher_name(), cipher_numeric_id(), cipher_vector_length(), client_connect(), client_read(), client_read_line(), client_write(), cmp_mt_mt(), compress_import(), config_free(), config_output_value(), config_output_value_

generic(), contact_alloc(), contact_create(), contact_delete(), contact_detail_alloc(), contact_detail_delete(), contact_detail_upsert(), contact_details_fetch(), contact_edit(), contact_find_number(), contact_folder_create(), contact_folder_remove(), contact_folder_rename(), contact_move(), contact_remove(), contact_update(), contact_update_stamp(), contacts_fetch(), contacts_update(), cryptex_alloc(), deprecated_cryptex_alloc(), deprecated_ecies_key_private_hex(), deprecated_hmac_digest(), deprecated_scramble_decrypt(), deprecated_scramble_encrypt(), deprecated_scramble_import(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), deprecated_symmetric_vector(), deserialize_int16(), deserialize_int32(), deserialize_int64(), deserialize_ns(), deserialize_st(), deserialize_uint16(), deserialize_uint32(), deserialize_uint64(), dh_exchange_2048(), dh_exchange_4096(), dh_params_generate(), digest_id(), digest_length_output(), digest_name(), dkim_signature_create(), dkim_signature_verify(), dkim_start(), domain_alloc(), double_conv(), dspam_check(), dspam_train(), ecies_key_private_hex(), ed25519_alloc(), ed25519_free(), ed25519_private_get(), ed25519_public_get(), ed25519_sign(), ed25519_verify(), encrypted_chunk_alloc(), encrypted_chunk_free(), encrypted_chunk_get(), encrypted_chunk_set(), encrypted_message_alloc(), encrypted_message_free(), engine_compress(), engine_decompress(), ephemeral_chunk_alloc(), ephemeral_chunk_free(), ephemeral_chunk_get(), ephemeral_chunk_set(), file_read(), file_temp_handle(), float_conv(), hash_digest(), hashed_bucket_get_ptr(), hashed_bucket_set_ptr(), hashed_cursor_alloc(), hex_decode_chr(), hex_decode_opts(), hex_decode_st(), hex_encode_opts(), hex_encode_st(), hmac_digest(), host_platform(), host_version(), http_body(), http_content_load_fonts(), http_content_start(), http_data_value_parse(), http_load_file(), http_page_get(), http_parse_context(), http_print_301(), http_process(), http_response_status(), ident_mt_mt(), imap_append(), imap_append_message(), imap_build_array(), imap_command_parser(), imap_copy(), imap_count_folder_levels(), imap_fetch_message(), imap_folder_create(), imap_folder_remove(), imap_folder_rename(), imap_folder_status(), imap_get_ar_ar(), imap_get_ptr(), imap_get_st_ar(), imap_get_type_ar(), imap_list(), imap_message_copier(), imap_narrow_folders(), imap_narrow_messages(), imap_parse_array(), imap_parse_astring(), imap_parse_literal(), imap_parse_qstring(), imap_range_build(), imap_search_messages_inner(), imap_starttls(), imap_valid_folder_name(), int16_clamp(), int16_conv_bl(), int32_clamp(), int32_conv_bl(), int64_clamp(), int64_conv_bl(), int8_clamp(), int8_conv_bl(), inx_append(), inx_count(), inx_delete(), inx_find(), inx_free(), inx_insert(), inx_options(), inx_replace(), inx_serial(), inx_truncate(), ip_presentation(), ip_reversed(), ip_standard(), ip_subnet(), json_api_dispatch(), keks_alloc(), keks_free(), lib_load_bzip(), line_pl_bl(), linked_append(), linked_cursor_alloc(), linked_record_free(), linked_record_get_data(), linked_record_get_key(), lock_get(), lock_release(), lower_st(), magma_folder_alloc(), magma_folder_fetch(), magma_folder_name(), mail_add_forward_headers(), mail_add_inbound_headers(), mail_add_outbound_headers(), mail_add_required_headers(), mail_build_signature(), mail_cache_set(), mail_cache_start(), mail_cache_stop(), mail_copy_message(), mail_count_received(), mail_create_message(), mail_db_insert_duplicate_message(), mail_db_insert_message(), mail_db_update_message_folder(), mail_extract_address(), mail_extract_tag(), mail_get_boundary(), mail_get_chunk(), mail_header_end(), mail_headers(), mail_insert_chunk_base64(), mail_insert_chunk_text(), mail_load_header(), mail_load_message(), mail_message(), mail_message_cleanup(), mail_message_path(), mail_mime_boundary(), mail_mime_child(), mail_mime_count(), mail_mime_encode_part(), mail_mime_generate_boundary(), mail_mime_get_smtp_envelope(), mail_mime_part(), mail_mime_split(), mail_mod_subject(), mail_modify_part(), mail_move_message(), mail_remove_message(), mail_signature_add(), mail_store_message(), message_alloc(), message_folder_create(), message_folder_remove(), messages_update(), meta_alloc(), meta_crypto_keys_create(), meta_data_acknowledge_alert(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), meta_data_fetch_shard(), meta_data_fetch_user(), meta_data_flags_add(), meta_data_flags_remove(), meta_data_flags_replace(), meta_data_insert_keys(), meta_data_insert_shard(), meta_data_update_log(), meta_folder_stats_tag_alloc(), meta_folders_name(), meta_folders_stats_tags(), meta_get(), meta_messages_copier(), meta_messages_mover(), meta_update_keys(), meta_update_realms(), meta_user_ref_add(), meta_user_ref_dec(), meta_user_ref_protocol_total(), mm_alloc(), mm_dupe(), mm_free(), mm_sec_alloc(), mm_sec_realloc(), mm_sec_start(), mm_wipe(), mt_dupe(), mutex_destroy(), mutex_init(), net_set_blocking(), net_set_buffer_length(), net_set_keepalive(), net_set_linger(), net_set_nodelay(), net_set_reuseable_address(), net_set_timeout(), ns_alloc(), ns_append(), ns_dupe(), ns_free(), ns_import(), ns_length_int(), ns_wipe(), org_key_alloc(), org_key_generate(), org_key_get(), org_key_set(), org_signet_alloc(), org_signet_generate(), org_signet_get(), org_signet_set(), part_decrypt(), part_encrypt(), pool_get_obj(), pool_get_status(), pool_set_obj(), pool_set_status(), pop_num_parse(), pop_pass_parse(), pop_starttls(), pop_top_parse(), pop_user_parse(), portal_config_entry(), portal_contact_details(), portal_endpoint(), portal_endpoint_ad(), portal_endpoint_alert_acknowledge(), portal_endpoint_alert_list(), portal_endpoint_aliases(), portal_endpoint_attachments_add(), portal_endpoint_attachments_remove(), portal_endpoint_auth(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_edit(), portal_endpoint_contacts_list(), portal_endpoint_contacts_load(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), portal_endpoint_error(), portal_endpoint_folders_add(), portal_endpoint_folders_list(), portal_endpoint_folders_remove(), portal_endpoint_folders_rename(), portal_endpoint_folders_tags(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_load(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_endpoint_messages_tags(), portal_endpoint_response(), portal_get_upload_attachment(), portal_message_attachments(), portal_message_body(), portal_message_header(), portal_message_info(), portal_message_meta(), portal_message_security(), portal_message_server(), portal_message_source(), portal_meta(), portal_parse_json_str_array(), portal_settings_changepass(), portal_settings_identity(), portal_smtp_create_data(), portal_upload(), portal_validate_request(), prime_alloc(), prime_field_size_max(), prime_field_write(), prime_free(), prime_get(), prime_header_length(), prime_header_read(), prime_header_write(), prime_key_decrypt(), prime_key_encrypt(), prime_key_generate(), prime_message_decrypt(), prime_message_encrypt(), prime_object_alloc(), prime_object_size_max(), prime_object_size_min(), prime_object_type(), prime_pem_begin(), prime_pem_end(), prime_pem_wrap(), prime_request-

generate(), prime_request_sign(), prime_set(), prime_signet_fingerprint(), prime_signet_generate(), prime_signet_validate(), process_find_pid(), process_kill(), process_stop(), protocol_enqueue(), protocol_process(), protocol_secure(), qp_decode(), qp_encode(), rand_choices(), rand_get_int16(), rand_get_int32(), rand_get_int64(), rand_get_int8(), rand_get_uint16(), rand_get_uint32(), rand_get_uint64(), rand_get_uint8(), rand_write(), register_business_step1(), register_business_step2(), register_captcha_generate(), register_captcha_random_font(), register_data_fetch_blocklist(), register_data_insert_user(), register_process(), register_session_cache(), register_session_generate(), register_session_get(), relay_free(), relay_output_settings(), relay_validate(), rwlock_attr_destroy(), rwlock_attr_getkind(), rwlock_attr_init(), rwlock_attr_setkind(), rwlock_destroy(), rwlock_init(), rwlock_lock_read(), rwlock_lock_write(), rwlock_unlock(), scramble_decrypt(), scramble_encrypt(), scramble_import(), secp256k1_compute_kek(), secp256k1_free(), secp256k1_private_get(), secp256k1_public_get(), serial_get(), serial_increment(), serial_prefix(), serial_reset(), servers_free(), servers_get_by_protocol(), servers_output_settings(), servers_validate(), sess_create(), sess_get(), sess_key(), sess_token(), signal_name(), signature_tree_alloc(), signature_tree_free(), slots_actors(), slots_alloc(), slots_free(), slots_get(), slots_key(), smtp_accept_message(), smtp_add_bypass_entry(), smtp_add_recipient(), smtp_bounce(), smtp_check_filters(), smtp_check_greylist(), smtp_check_rbl(), smtp_check_receive_quota(), smtp_check_transmit_quota(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), smtp_client_send_nullfrom(), smtp_client_send_rcptto(), smtp_data_read(), smtp_fetch_authorization(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_fetch_rollmessages(), smtp_forward_message(), smtp_insert_spamsig(), smtp_parse_auth(), smtp_parse_helo_domain(), smtp_parse_mail_from_path(), smtp_parse_rcpt_to(), smtp_relay_message(), smtp_reply(), smtp_rollout(), smtp_send_message(), smtp_starttls(), smtp_store_message(), smtp_store_spamsig(), smtp_update_receive_stats(), smtp_update_transmission_stats(), spf_check(), spf_start(), spool_cleanup(), spool_mktemp(), spool_stop(), sql_insert(), sql_insert_conn(), sql_num_rows(), sql_num_rows_conn(), sql_query_conn(), sql_query_res(), sql_query_res_conn(), sql_write(), sql_write_conn(), ssl_error_string(), ssl_verify_privkey(), st_alloc_opts(), st_append_opts(), st_append_out(), st_avail_get(), st_avail_set(), st_bitwise(), st_copy_in(), st_data_get(), st_data_set(), st_dupe_opts(), st_free(), st_import_opts(), st_length_get(), st_length_int(), st_length_set(), st_merge_opts(), st_not(), st_opt_set(), st_opt_test(), st_output(), st_realloc(), st_replace(), st_search_chr(), st_search_ci(), st_search_cs(), st_set(), st_swap(), st_vaprint_opts(), st_vsprint(), st_wipe(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_decrypt(), stacie_derive_key(), stacie_derive_rounds(), stacie_derive_seed(), stacie_derive_token(), stacie_encrypt(), stacie_realm_key(), stacker_alloc(), stacker_push(), stamp_counter_check(), stamp_counter_increment(), statistics_process(), statistics_refresh(), stats_derived_value(), status_signal(), stmt_bind_param(), stmt_reset(), symmetric_decrypt(), symmetric_encrypt(), symmetric_vector(), system_ulimit_cur(), system_ulimit_max(), tank_count(), tank_delete_object(), tank_load(), tcp_continue(), tcp_error(), tcp_read(), tcp_write(), teacher_add_cookie(), teacher_data_delete(), teacher_data_get(), teacher_data_save(), thread_alloc(), thread_cancel(), thread_join(), thread_launch(), thread_result(), time_print_gmt(), time_print_local(), tkey_init(), tkey_set(), tls_client_alloc(), tls_continue(), tls_error(), tls_free(), tls_read(), tls_server_alloc(), tls_server_destroy(), tls_write(), tok_get_count_bl(), tok_get_ns(), tree_alloc(), tree_cursor_alloc(), uint16_clamp(), uint16_conv_bl(), uint16_get_no(), uint16_put_no(), uint24_get_no(), uint24_put_no(), uint32_clamp(), uint32_conv_bl(), uint32_get_no(), uint32_put_no(), uint64_clamp(), uint64_conv_bl(), uint8_clamp(), uint8_conv_bl(), upper_st(), url_decode(), url_encode(), user_config_alloc(), user_config_create(), user_config_delete(), user_config_edit(), user_config_entry_alloc(), user_config_fetch(), user_config_upsert(), user_key_alloc(), user_key_generate(), user_key_get(), user_key_set(), user_request_generate(), user_request_get(), user_request_rotation(), user_request_set(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_alloc(), user_signet_get(), user_signet_set(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), user_signet_verify_self(), utf8_length_st(), utf8_valid_st(), virus_check(), virus_start(), warehouse_fetch_domains(), warehouse_fetch_patterns(), xml_create_doc(), xml_create_parser_ctx(), xml_create_xpath_ctx(), xml_free_doc(), xml_free_parser_ctx(), xml_free_xpath_ctx(), xml_free_xpath_obj(), xml_get_xpath_int16(), xml_get_xpath_int32(), xml_get_xpath_int64(), xml_get_xpath_int8(), xml_get_xpath_node_count(), xml_get_xpath_ns(), xml_get_xpath_st(), xml_get_xpath_uint16(), xml_get_xpath_uint32(), xml_get_xpath_uint64(), xml_get_xpath_uint8(), xml_node_get_content_st(), xml_set_xpath_ns(), xml_set_xpath_property(), xml_set_xpath_uint64(), xml_xpath_eval(), zbase32_decode(), and zbase32_encode().

5.18.1.10 #define SIGUNUSED 31

If we encounter a system that doesn't define SIGUNUSED, we define it.

Definition at line 137 of file core.h.

Referenced by signal_name().

5.18.1.11 #define UINT24_MAX (16777215)

Definition at line 116 of file core.h.

5.18.1.12 `#define UINT24_MIN (0)`

Definition at line 112 of file core.h.

5.18.2 Typedef Documentation

5.18.2.1 `typedef char bool_t`

Definition at line 46 of file core.h.

5.18.2.2 `typedef unsigned char byte_t`

Definition at line 65 of file core.h.

5.18.2.3 `typedef char chr_t`

Definition at line 53 of file core.h.

5.18.2.4 `typedef __int24_t int24_t`

Definition at line 88 of file core.h.

5.18.2.5 `typedef int32_t int_t`

Definition at line 71 of file core.h.

5.18.2.6 `typedef unsigned char uchr_t`

Definition at line 59 of file core.h.

5.18.2.7 `typedef __uint24_t uint24_t`

Definition at line 99 of file core.h.

5.18.2.8 `typedef uint32_t uint_t`

Definition at line 77 of file core.h.

5.18.3 Enumeration Type Documentation

5.18.3.1 anonymous enum

Enumerator:

EMPTY

Definition at line 164 of file core.h.

5.18.3.2 enum M_TYPE

Different types used throughout.

Enumerator:

M_TYPE_EMPTY M_TYPE_EMPTY.
M_TYPE_MULTI M_TYPE_MULTI is [multi_t](#).
M_TYPE_ENUM M_TYPE_ENUM is enum.
M_TYPE_BOOLEAN M_TYPE_BOOLEAN is bool_t.
M_TYPE_BLOCK M_TYPE_BLOCK is void pointer.
M_TYPE_NULLER M_TYPE_NULLER is char pointer.
M_TYPE_PLACER M_TYPE_PLACER is placer_t struct.
M_TYPE_STRINGER M_TYPE_STRINGER is stringer_t pointer.
M_TYPE_INT8 M_TYPE_INT8 is int8_t.
M_TYPE_INT16 M_TYPE_INT16 is int16_t.
M_TYPE_INT32 M_TYPE_INT32 is int32_t.
M_TYPE_INT64 M_TYPE_INT64 is int64_t.
M_TYPE_UINT8 M_TYPE_UINT8 is uint8_t.
M_TYPE_UINT16 M_TYPE_UINT16 is uint16_t.
M_TYPE_UINT32 M_TYPE_UINT32 is uint32_t.
M_TYPE_UINT64 M_TYPE_UINT64 is uint64_t.
M_TYPE_FLOAT M_TYPE_FLOAT is float.
M_TYPE_DOUBLE M_TYPE_DOUBLE is double.

Definition at line 143 of file core.h.

5.18.4 Function Documentation

5.18.4.1 struct __attribute__((packed)) [read]

Definition at line 94 of file core.h.

5.18.4.2 char* type (M_TYPE type)

Takes a type code and returns the fully enumerated string associated with that type.

Parameters:

type The type code to evaluate.

Returns:

Null terminated string containing type name. String is stored in static buffer and returned as a pointer.

Definition at line 17 of file type.c.

References M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_ENUM, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_MULTI, M_TYPE_NULLER, M_TYPE_PLACER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, and M_TYPE_UINT8.

Referenced by `__attribute__()`, `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `ar_dupe()`, `ar_free()`, `cache_free()`, `cache_output_settings()`, `cache_set_value()`, `cache_validate()`, `chunk_buffer_size()`, `chunk_header_size()`, `chunk_header_type()`, `config_free()`, `encrypted_chunk_get()`, `imap_parse_array()`, `imap_parse_dataitems()`, `mail_modify_part()`, `meta_data_delete_folder()`, `meta_data_fetch_folders()`, `meta_data_insert_folder()`, `meta_data_update_folder_name()`, `mt_dupe()`, `mt_get_char()`, `mt_get_length()`, `mt_get_number()`, `mt_get_type()`, `mt_is_empty()`, `mt_is_number()`, `mt_set_type()`, `naked_message_get()`, `naked_message_set()`, `part_decrypt()`, `portal_message_attachments()`, `portal_message_body()`, `prime_key_decrypt()`, `prime_pem_unwrap()`, `prime_pem_wrap()`, `prime_set()`, `prime_unpack()`, `prime_unpack_fields()`, `prime_unpack_validate()`, `relay_free()`, `relay_output_settings()`, `relay_set_value()`, `relay_validate()`, `servers_free()`, `servers_get_by_protocol()`, `servers_output_settings()`, `servers_set_value()`, `servers_validate()`, `signature_full_verify()`, and `signature_tree_get()`.

5.18.5 Variable Documentation

5.18.5.1 `__int24_t`

Definition at line 87 of file `core.h`.

5.18.5.2 `__uint24_t`

Definition at line 98 of file `core.h`.

5.18.5.3 `bool_t log_enabled`

Definition at line 11 of file `log.c`.

Referenced by `log_disable()`, `log_enable()`, and `log_internal()`.

5.18.5.4 `pthread_mutex_t log_mutex`

Definition at line 13 of file `log.c`.

Referenced by `log_disable()`, `log_enable()`, `log_internal()`, and `log_rotate()`.

5.19 src/core/encodings/base64.c File Reference

```
#include "magma.h"
```

Functions

- `size_t base64_encoded_length_mod (size_t length)`
TODO: Switch to always using the "_wrap" length function, which can then become base64_(en|de)coded_length().
- `size_t base64_encoded_length (size_t length)`
- `size_t base64_encoded_length_wrap (size_t length, size_t wrap, base64_wrap_t type)`
Calculate the length of a binary buffer after being encoded with using base64.
- `size_t base64_decoded_length_mod (size_t length)`
- `size_t base64_decoded_length (size_t length)`
- `stringer_t * base64_encode (stringer_t *s, stringer_t *output)`
Perform base64 encoding on a managed string with padding and line splitting at BASE64_LINE_WRAP_LENGTH characters.
- `stringer_t * base64_encode_wrap (stringer_t *s, size_t wrap, base64_wrap_t type, stringer_t *output)`
Encode a binary buffer using base64, with a configurable line length, and terminating character.
- `stringer_t * base64_encode_mod (stringer_t *s, stringer_t *output)`
Perform modified base64 encoding on a managed string without padding or line splitting.
- `stringer_t * base64_encode_opts (stringer_t *s, uint32_t opts, bool_t modified)`
Perform base64 encoding on a managed string with padding and line splitting at BASE64_LINE_WRAP_LENGTH characters.
- `stringer_t * base64_decode (stringer_t *s, stringer_t *output)`
Perform base64 decoding on a managed string.
- `stringer_t * base64_decode_mod (stringer_t *s, stringer_t *output)`
Perform modified base64 decoding on a managed string. without padding or line splitting.
- `stringer_t * base64_decode_opts (stringer_t *s, uint32_t opts, bool_t modified)`
Perform base64 decoding on a managed string.

5.19.1 Function Documentation

5.19.1.1 stringer_t* base64_decode (stringer_t * s, stringer_t * output)

Perform base64 decoding on a managed string. [base64.c](#)

Parameters:

- `s` the managed string to be base64 decoded.
- output*** a managed string to receive the encoded output; if passed as NULL, one will be allocated to the caller.

Returns:

NULL on failure, or a newly allocated managed string containing the decoded result on success.

Definition at line 536 of file base64.c.

References mappings_t::base64, base64_decoded_length(), log_pedantic, mappings, st_alloc(), st_avail_get(), st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), st_valid_tracked(), and mappings_t::values.

Referenced by base64_decode_opts(), mail_insert_chunk_base64(), prime_pem_unwrap(), smtp_auth_login(), and smtp_auth_plain().

5.19.1.2 stringer_t* base64_decode_mod (stringer_t * s, stringer_t * output)

Perform modified base64 decoding on a managed string. without padding or line splitting.

Note:

In this function, the '+' and '/' characters are replaced with '-' and '_' respectively, making the output suitable for use in URL parameters without additional encoding.

Parameters:

s the managed string to be base64 decoded.

output a managed string to receive the encoded output; if passed as NULL, one will be allocated to the caller.

Returns:

NULL on failure, or a newly allocated managed string containing the modified decoded result on success.

Definition at line 636 of file base64.c.

References base64_decoded_length_mod(), mappings_t::base64_mod, log_pedantic, mappings, st_alloc(), st_avail_get(), st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), st_valid_tracked(), and mappings_t::values.

Referenced by auth_data_fetch(), base64_decode_opts(), meta_data_fetch_keys(), meta_data_fetch_shard(), meta_data_fetch_user(), and smtp_fetch_inbound().

5.19.1.3 stringer_t* base64_decode_opts (stringer_t * s, uint32_t opts, bool_t modified)

Perform base64 decoding on a managed string.

Parameters:

s the managed string to be base64 decoded.

opts the options value of the managed string which will be allocated and receive the decoded output.

modified if true, use modified base64 encoding; if false, use normal base64 decoding.

Returns:

NULL on failure, or a pointer to a newly allocated managed string containing the decoded result on success.

Definition at line 734 of file base64.c.

References base64_decode(), base64_decode_mod(), base64_decoded_length(), base64_decoded_length_mod(), log_pedantic, st_alloc_opts(), st_empty_out(), and st_free().

5.19.1.4 size_t base64_decoded_length (size_t length)

Definition at line 130 of file base64.c.

Referenced by base64_decode(), and base64_decode_opts().

5.19.1.5 `size_t base64_decoded_length_mod (size_t length)`

Definition at line 111 of file base64.c.

Referenced by `base64_decode_mod()`, and `base64_decode_opts()`.

5.19.1.6 `stringer_t* base64_encode (stringer_t * s, stringer_t * output)`

Perform base64 encoding on a managed string with padding and line splitting at `BASE64_LINE_WRAP_LENGTH` characters.

Parameters:

s the managed string to be base64 encoded.

output a managed string to receive the encoded output; if passed as `NULL`, one will be allocated to the caller.

Returns:

`NULL` on failure, or a pointer to the managed string containing the encoded result on success.

Definition at line 156 of file base64.c.

References `mappings_t::base64`, `base64_encoded_length()`, `BASE64_LINE_WRAP_LENGTH`, `mappings_t::characters`, `log_error`, `log_pedantic`, `mappings`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `base64_encode_opts()`, `mail_insert_chunk_base64()`, and `mail_mime_encode_part()`.

5.19.1.7 `stringer_t* base64_encode_mod (stringer_t * s, stringer_t * output)`

Perform modified base64 encoding on a managed string without padding or line splitting.

Note:

In this function, the `'+'` and `'/'` characters are replaced with `'-'` and `'_'` respectively, making the output suitable for use in URL parameters without additional encoding.

Parameters:

s the managed string to be base64 modified-encoded.

output a managed string to receive the encoded output; if passed as `NULL`, an output buffer will be allocated which must be freed by the caller.

Returns:

`NULL` on failure, or a newly allocated managed string containing the modified encoded result on success.

Definition at line 412 of file base64.c.

References `base64_encoded_length_mod()`, `mappings_t::base64_mod`, `mappings_t::characters`, `log_pedantic`, `mappings`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `auth_login()`, `base64_encode_opts()`, `meta_data_insert_keys()`, `meta_data_insert_shard()`, `register_data_insert_user()`, and `smtp_fetch_authorization()`.

5.19.1.8 `stringer_t* base64_encode_opts (stringer_t * s, uint32_t opts, bool_t modified)`

Perform base64 encoding on a managed string with padding and line splitting at `BASE64_LINE_WRAP_LENGTH` characters.

Parameters:

s the managed string to be base64 encoded.

opts the options value of the managed string which will be allocated and receive the encoded output.

modified if true, use modified base64 encoding; if false, use normal base64 encoding.

Returns:

NULL on failure, or a pointer to a newly allocated managed string containing the encoded result on success.

Definition at line 496 of file base64.c.

References base64_encode(), base64_encode_mod(), base64_encoded_length(), base64_encoded_length_mod(), log_pedantic, st_alloc_opts(), st_empty_out(), and st_free().

5.19.1.9 stringer_t* base64_encode_wrap (stringer_t * s, size_t wrap, base64_wrap_t type, stringer_t * output)

Encode a binary buffer using base64, with a configurable line length, and terminating character.

Parameters:

s the managed string to be base64 encoded.

wrap the maximum length of each line, or 0 to disable line wrapping.

type the line delimiter sequence being used.

output a managed string to receive the encoded output; if passed as NULL, one will be allocated to the caller. *

Returns:

Definition at line 271 of file base64.c.

References mappings_t::base64, base64_encoded_length_wrap(), BASE64_LINE_WRAP_CRLF, BASE64_LINE_WRAP_LF, mappings_t::characters, log_error, log_pedantic, mappings, st_alloc(), st_avail_get(), st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), and st_valid_tracked().

Referenced by prime_pem_wrap().

5.19.1.10 size_t base64_encoded_length (size_t length)

Definition at line 34 of file base64.c.

References BASE64_LINE_WRAP_LENGTH.

Referenced by base64_encode(), and base64_encode_opts().

5.19.1.11 size_t base64_encoded_length_mod (size_t length)

TODO: Switch to always using the "_wrap" length function, which can then become base64_(en|de)coded_length().

Definition at line 12 of file base64.c.

Referenced by base64_encode_mod(), and base64_encode_opts().

5.19.1.12 size_t base64_encoded_length_wrap (size_t length, size_t wrap, base64_wrap_t type)

Calculate the length of a binary buffer after being encoded with using base64.

Note:

If line wrapping is enabled, the output will assume a trailing end of line sequence.

Parameters:

- length* the length of the binary buffer.
- wrap* the maximum length of each line, or 0 to disable line wrapping.
- type* the line delimiter sequence being used.

Returns:

- the length of the encoded output.

Definition at line 71 of file base64.c.

References BASE64_LINE_WRAP_NONE, and log_pedantic.

Referenced by base64_encode_wrap(), and prime_pem_wrap().

5.20 src/core/encodings/encodings.h File Reference

Data Structures

- struct [mappings_t](#)

Defines

- #define [URL_MAX_LENGTH](#) 1048576
- #define [QP_LINE_WRAP_LENGTH](#) 76
- #define [BASE64_LINE_WRAP_LENGTH](#) 76

Enumerations

- enum [base64_wrap_t](#) { [BASE64_LINE_WRAP_NONE](#) = 0, [BASE64_LINE_WRAP_LF](#) = 1, [BASE64_LINE_WRAP_CRLF](#) = 2 }

Functions

- [stringer_t * base64_decode](#) ([stringer_t](#) *s, [stringer_t](#) *output)
base64.c
- [stringer_t * base64_decode_mod](#) ([stringer_t](#) *s, [stringer_t](#) *output)
Perform modified base64 decoding on a managed string. without padding or line splitting.
- [stringer_t * base64_decode_opts](#) ([stringer_t](#) *s, [uint32_t](#) opts, [bool_t](#) modified)
Perform base64 decoding on a managed string.
- [size_t base64_decoded_length](#) ([size_t](#) length)
- [size_t base64_decoded_length_mod](#) ([size_t](#) length)
- [stringer_t * base64_encode](#) ([stringer_t](#) *s, [stringer_t](#) *output)
Perform base64 encoding on a managed string with padding and line splitting at [BASE64_LINE_WRAP_LENGTH](#) characters.
- [stringer_t * base64_encode_mod](#) ([stringer_t](#) *s, [stringer_t](#) *output)
Perform modified base64 encoding on a managed string without padding or line splitting.
- [stringer_t * base64_encode_opts](#) ([stringer_t](#) *s, [uint32_t](#) opts, [bool_t](#) modified)
Perform base64 encoding on a managed string with padding and line splitting at [BASE64_LINE_WRAP_LENGTH](#) characters.
- [stringer_t * base64_encode_wrap](#) ([stringer_t](#) *s, [size_t](#) wrap, [base64_wrap_t](#) type, [stringer_t](#) *output)
Encode a binary buffer using base64, with a configurable line length, and terminating character.
- [size_t base64_encoded_length](#) ([size_t](#) length)
- [size_t base64_encoded_length_mod](#) ([size_t](#) length)
TODO: Switch to always using the "_wrap" length function, which can then become base64_(en|de)coded_length().
- [size_t base64_encoded_length_wrap](#) ([size_t](#) length, [size_t](#) wrap, [base64_wrap_t](#) type)
Calculate the length of a binary buffer after being encoded with using base64.
- [bool_t hex_valid_chr](#) ([uchr_t](#) c)
hex.c
- [byte_t hex_decode_chr](#) ([uchr_t](#) a, [uchr_t](#) b)

Decode a hex character pair as a single byte (usually for URL decoding).

- `size_t hex_count_st (stringer_t *s)`
Count the number of hex characters in a string.
- `size_t hex_valid_st (stringer_t *s)`
Determine whether a managed string is a properly formatted hex string and return the number found.
- `stringer_t * hex_decode_st (stringer_t *h, stringer_t *output)`
Convert a hex string into a binary data blob.
- `stringer_t * hex_encode_st (stringer_t *b, stringer_t *output)`
Convert a block of binary data into a hex string.
- `uchr_t * hex_encode_chr (byte_t b, uchr_t *output)`
- `stringer_t * hex_encode_st_debug (stringer_t *input, size_t maxlen)`
Encode data in a human readable way, for debugging purposes.
- `stringer_t * hex_encode_opts (stringer_t *input, uint32_t opts)`
Allocates an output string of appropriate size with specified opts for hex encoding of input.
- `stringer_t * hex_decode_opts (stringer_t *input, uint32_t opts)`
Allocates an output string of appropriate size with specified opts for hex decoding of input.
- `stringer_t * qp_decode (stringer_t *s)`
qp.c
- `stringer_t * qp_encode (stringer_t *s)`
Perform QP (quoted-printable) encoding of a string.
- `bool_t url_valid_chr (uchr_t c)`
url.c
- `size_t url_valid_st (stringer_t *s)`
Check a URL string for validity.
- `stringer_t * url_decode (stringer_t *s)`
Decode a URL-encoded string into its original representation.
- `stringer_t * url_encode (stringer_t *s)`
Encode a data buffer as a valid URL component.
- `stringer_t * zbase32_decode (stringer_t *s)`
zbase32.c
- `stringer_t * zbase32_encode (stringer_t *s)`
Encode data as a zbase32 string.

Variables

- `mappings_t mappings`

5.20.1 Define Documentation

5.20.1.1 `#define BASE64_LINE_WRAP_LENGTH 76`

Definition at line 24 of file encodings.h.

Referenced by `base64_encode()`, and `base64_encoded_length()`.

5.20.1.2 `#define QP_LINE_WRAP_LENGTH 76`

Definition at line 23 of file encodings.h.

Referenced by `qp_encode()`.

5.20.1.3 `#define URL_MAX_LENGTH 1048576`

Note:

When considering a change to the URL length limit, the following statistics may be useful:

Internet Explorer: 2,048 characters for the host/path and 2,083 characters overall
Firefox: limited to 65,536 visible characters, but longer URL will still work
Safari: at least 80,000 characters
Opera: at least 190,000 characters
IIS: by default the limit is 16,384 but can be increased
Apache: by default 4,000 characters

Definition at line 22 of file encodings.h.

5.20.2 Enumeration Type Documentation

5.20.2.1 `enum base64_wrap_t`

Enumerator:

BASE64_LINE_WRAP_NONE

BASE64_LINE_WRAP_LF

BASE64_LINE_WRAP_CRLF

Definition at line 26 of file encodings.h.

5.20.3 Function Documentation

5.20.3.1 `stringer_t* base64_decode (stringer_t * s, stringer_t * output)`

[base64.c](#) [base64.c](#)

Parameters:

s the managed string to be base64 decoded.

output a managed string to receive the encoded output; if passed as NULL, one will be allocated to the caller.

Returns:

NULL on failure, or a newly allocated managed string containing the decoded result on success.

Definition at line 536 of file base64.c.

References `mappings_t::base64`, `base64_decoded_length()`, `log_pedantic`, `mappings`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, `st_valid_tracked()`, and `mappings_t::values`.

Referenced by `base64_decode_opts()`, `mail_insert_chunk_base64()`, `prime_pem_unwrap()`, `smtp_auth_login()`, and `smtp_auth_plain()`.

5.20.3.2 `stringer_t* base64_decode_mod (stringer_t * s, stringer_t * output)`

Perform modified base64 decoding on a managed string. without padding or line splitting.

Note:

In this function, the '+' and '/' characters are replaced with '-' and '_' respectively, making the output suitable for use in URL parameters without additional encoding.

Parameters:

s the managed string to be base64 decoded.

output a managed string to receive the encoded output; if passed as NULL, one will be allocated to the caller.

Returns:

NULL on failure, or a newly allocated managed string containing the modified decoded result on success.

Definition at line 636 of file base64.c.

References `base64_decoded_length_mod()`, `mappings_t::base64_mod`, `log_pedantic`, `mappings`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, `st_valid_tracked()`, and `mappings_t::values`.

Referenced by `auth_data_fetch()`, `base64_decode_opts()`, `meta_data_fetch_keys()`, `meta_data_fetch_shard()`, `meta_data_fetch_user()`, and `smtp_fetch_inbound()`.

5.20.3.3 `stringer_t* base64_decode_opts (stringer_t * s, uint32_t opts, bool_t modified)`

Perform base64 decoding on a managed string.

Parameters:

s the managed string to be base64 decoded.

opts the options value of the managed string which will be allocated and receive the decoded output.

modified if true, use modified base64 encoding; if false, use normal base64 decoding.

Returns:

NULL on failure, or a pointer to a newly allocated managed string containing the decoded result on success.

Definition at line 734 of file base64.c.

References `base64_decode()`, `base64_decode_mod()`, `base64_decoded_length()`, `base64_decoded_length_mod()`, `log_pedantic`, `st_alloc_opts()`, `st_empty_out()`, and `st_free()`.

5.20.3.4 `size_t base64_decoded_length (size_t length)`

Definition at line 130 of file base64.c.

Referenced by `base64_decode()`, and `base64_decode_opts()`.

5.20.3.5 `size_t base64_decoded_length_mod (size_t length)`

Definition at line 111 of file base64.c.

Referenced by `base64_decode_mod()`, and `base64_decode_opts()`.

5.20.3.6 stringer_t* base64_encode (stringer_t * s, stringer_t * output)

Perform base64 encoding on a managed string with padding and line splitting at BASE64_LINE_WRAP_LENGTH characters.

Parameters:

s the managed string to be base64 encoded.

output a managed string to receive the encoded output; if passed as NULL, one will be allocated to the caller.

Returns:

NULL on failure, or a pointer to the managed string containing the encoded result on success.

Definition at line 156 of file base64.c.

References mappings_t::base64, base64_encoded_length(), BASE64_LINE_WRAP_LENGTH, mappings_t::characters, log_error, log_pedantic, mappings, st_alloc(), st_avail_get(), st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), and st_valid_tracked().

Referenced by base64_encode_opts(), mail_insert_chunk_base64(), and mail_mime_encode_part().

5.20.3.7 stringer_t* base64_encode_mod (stringer_t * s, stringer_t * output)

Perform modified base64 encoding on a managed string without padding or line splitting.

Note:

In this function, the '+' and '/' characters are replaced with '-' and '_' respectively, making the output suitable for use in URL parameters without additional encoding.

Parameters:

s the managed string to be base64 modified-encoded.

output a managed string to receive the encoded output; if passed as NULL, an output buffer will be allocated which must be freed by the caller.

Returns:

NULL on failure, or a newly allocated managed string containing the modified encoded result on success.

Definition at line 412 of file base64.c.

References base64_encoded_length_mod(), mappings_t::base64_mod, mappings_t::characters, log_pedantic, mappings, st_alloc(), st_avail_get(), st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), and st_valid_tracked().

Referenced by auth_login(), base64_encode_opts(), meta_data_insert_keys(), meta_data_insert_shard(), register_data_insert_user(), and smtp_fetch_authorization().

5.20.3.8 stringer_t* base64_encode_opts (stringer_t * s, uint32_t opts, bool_t modified)

Perform base64 encoding on a managed string with padding and line splitting at BASE64_LINE_WRAP_LENGTH characters.

Parameters:

s the managed string to be base64 encoded.

opts the options value of the managed string which will be allocated and receive the encoded output.

modified if true, use modified base64 encoding; if false, use normal base64 encoding.

Returns:

NULL on failure, or a pointer to a newly allocated managed string containing the encoded result on success.

Definition at line 496 of file base64.c.

References `base64_encode()`, `base64_encode_mod()`, `base64_encoded_length()`, `base64_encoded_length_mod()`, `log_pedantic`, `st_alloc_opts()`, `st_empty_out()`, and `st_free()`.

5.20.3.9 `stringer_t* base64_encode_wrap (stringer_t * s, size_t wrap, base64_wrap_t type, stringer_t * output)`

Encode a binary buffer using base64, with a configurable line length, and terminating character.

Parameters:

s the managed string to be base64 encoded.

wrap the maximum length of each line, or 0 to disable line wrapping.

type the line delimiter sequence being used.

output a managed string to receive the encoded output; if passed as NULL, one will be allocated to the caller. *

Returns:

Definition at line 271 of file base64.c.

References `mappings_t::base64`, `base64_encoded_length_wrap()`, `BASE64_LINE_WRAP_CRLF`, `BASE64_LINE_WRAP_LF`, `mappings_t::characters`, `log_error`, `log_pedantic`, `mappings`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `prime_pem_wrap()`.

5.20.3.10 `size_t base64_encoded_length (size_t length)`

Definition at line 34 of file base64.c.

References `BASE64_LINE_WRAP_LENGTH`.

Referenced by `base64_encode()`, and `base64_encode_opts()`.

5.20.3.11 `size_t base64_encoded_length_mod (size_t length)`

TODO: Switch to always using the "_wrap" length function, which can then become `base64_(en|de)coded_length()`.

Definition at line 12 of file base64.c.

Referenced by `base64_encode_mod()`, and `base64_encode_opts()`.

5.20.3.12 `size_t base64_encoded_length_wrap (size_t length, size_t wrap, base64_wrap_t type)`

Calculate the length of a binary buffer after being encoded with using base64.

Note:

If line wrapping is enabled, the output will assume a trailing end of line sequence.

Parameters:

length the length of the binary buffer.

wrap the maximum length of each line, or 0 to disable line wrapping.

type the line delimiter sequence being used.

Returns:

the length of the encoded output.

Definition at line 71 of file base64.c.

References BASE64_LINE_WRAP_NONE, and log_pedantic.

Referenced by base64_encode_wrap(), and prime_pem_wrap().

5.20.3.13 size_t hex_count_st (stringer_t * s)

Count the number of hex characters in a string.

Parameters:

s the managed string to be scanned.

Returns:

the number of valid hexadecimal characters found in the string.

Definition at line 58 of file hex.c.

References hex_valid_chr(), and st_empty_out().

Referenced by hex_decode_st().

5.20.3.14 byte_t hex_decode_chr (uchr_t a, uchr_t b)

Decode a hex character pair as a single byte (usually for URL decoding).

Parameters:

a the higher order 4-bit hexadecimal character of the byte to be encoded.

b the lower order 4-bit hexadecimal character of the byte to be decoded.

Returns:

a byte containing the value of the decoded hexadecimal character pair, or 0 on failure.

Definition at line 252 of file hex.c.

References hex_valid_chr(), log_pedantic, and lower_chr().

Referenced by hex_decode_st(), http_data_value_decode(), qp_decode(), and url_decode().

5.20.3.15 stringer_t* hex_decode_opts (stringer_t * input, uint32_t opts)

Allocates an output string of appropriate size with specified opts for hex decoding of input.

Parameters:

input Input stringer to be decoded.

Returns:

NULL on failure, otherwise allocated stringer with decoded data.

Definition at line 383 of file hex.c.

References hex_decode_st(), log_error, log_pedantic, st_alloc_opts(), st_empty, st_free(), and st_length_get().

5.20.3.16 stringer_t* hex_decode_st (stringer_t * *h*, stringer_t * *output*)

Convert a hex string into a binary data blob.

Note:

All hex strings should be composed of pairs of two hex characters representing individual bytes. Invalid hex characters will simply be ignored during processing.

Parameters:

h a managed string containing the input hex string to be decoded.

output if not NULL, a pointer to a managed string to contain the decoded binary output; if NULL, a new string will be allocated and returned to the caller.

Returns:

a pointer to a managed string containing the decoded output, or NULL on failure.

Definition at line 287 of file hex.c.

References hex_count_st(), hex_decode_chr(), hex_valid_chr(), log_pedantic, st_alloc(), st_avail_get(), st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), and st_valid_tracked().

Referenced by auth_data_fetch(), and hex_decode_opts().

5.20.3.17 uchr_t* hex_encode_chr (byte_t *b*, uchr_t * *output*)

Converts a binary octet into a pair of lowercase hex characters. The result is returned inside a thread specific static character buffer. If a valid pointer is also supplied using the output variable then the result will also be written to the memory the pointer indicates.

Parameters:

b The binary octet being encoded.

output A pointer to the memory where the result should be stored or NULL if the result should only be returned.

Returns:

Returns the two character hex representation of the binary input using a thread localized character buffer.

Definition at line 84 of file hex.c.

References mm_wipe(), and number.

Referenced by hex_encode_st(), and hex_encode_st_debug().

5.20.3.18 stringer_t* hex_encode_opts (stringer_t * *input*, uint32_t *opts*)

Allocates an output string of appropriate size with specified opts for hex encoding of input.

Parameters:

input Input stringer to be encoded.

Returns:

NULL on failure, otherwise allocated stringer with encoded data.

Definition at line 346 of file hex.c.

References hex_encode_st(), log_error, log_pedantic, st_alloc_opts(), st_empty, st_free(), and st_length_get().

5.20.3.19 stringer_t* hex_encode_st (stringer_t * *b*, stringer_t * *output*)

Convert a block of binary data into a hex string.

Parameters:

b a managed string containing the raw data to be encoded.

output if not NULL, a pointer to a managed string that will store the encoded output; if NULL, a new managed string will be allocated and returned to the caller.

Returns:

NULL on failure, or a pointer to a managed string containing the hex-encoded output on success.

Definition at line 124 of file hex.c.

References `hex_encode_chr()`, `log_pedantic`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `auth_login()`, and `hex_encode_opts()`.

5.20.3.20 stringer_t* hex_encode_st_debug (stringer_t * *input*, size_t *maxlen*)

Encode data in a human readable way, for debugging purposes.

Note:

All printable ASCII data will be displayed as-is; all other characters will be shown as formatted hex pairs.

Parameters:

input a pointer to a managed string containing the data to be formatted.

maxlen the maximum number of bytes to be displayed. If more characters are available, an ellipsis will be shown in the middle of the string to represent the abridged data, with only the leading and trailing characters returned as part of the display string.

Returns:

NULL on failure, or a pointer to a newly allocated managed string containing the encoded debug data on success.

Definition at line 178 of file hex.c.

References `chr_printable()`, `hex_encode_chr()`, `log_error`, `st_alloc()`, `st_char_get()`, `st_length_get()`, and `st_length_set()`.

5.20.3.21 bool_t hex_valid_chr (uchar_t *c*)

[hex.c](#) [hex.c](#)

Parameters:

c the character to be tested.

Returns:

true if the character is a valid hexadecimal character; false otherwise.

Definition at line 15 of file hex.c.

Referenced by `hex_count_st()`, `hex_decode_chr()`, `hex_decode_st()`, `hex_valid_st()`, `qp_decode()`, `url_decode()`, and `url_valid_st()`.

5.20.3.22 `size_t hex_valid_st (stringer_t * s)`

Determine whether a managed string is a properly formatted hex string and return the number found.

Note:

A valid hex string consists of only hexadecimal characters and whitespace, and the number of hex characters is divisible by two.

Parameters:

s the managed string to be tested.

Returns:

0 on failure, or the number of hexadecimal characters found in the managed string.

Definition at line 28 of file hex.c.

References `hex_valid_chr()`, and `st_empty_out()`.

5.20.3.23 `stringer_t* qp_decode (stringer_t * s)`

[qp.c](#) `qp.c`

Parameters:

s the managed string containing data to be decoded.

Returns:

a pointer to a managed string containing the 8-bit decoded output, or NULL on failure.

Definition at line 99 of file qp.c.

References `hex_decode_chr()`, `hex_valid_chr()`, `log_pedantic`, `st_alloc()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, and `st_length_set()`.

5.20.3.24 `stringer_t* qp_encode (stringer_t * s)`

Perform QP (quoted-printable) encoding of a string.

Parameters:

s a pointer to a managed string containing data to be encoded.

Returns:

a pointer to a managed string containing the QP encoded data, or NULL on failure.

Definition at line 17 of file qp.c.

References `HEAP`, `JOINTED`, `log_pedantic`, `MANAGED_T`, `PLACER`, `QP_LINE_WRAP_LENGTH`, `st_alloc_opts()`, `st_append`, `st_data_get()`, and `st_empty_out()`.

Referenced by `mail_build_signature()`, and `mail_mime_encode_part()`.

5.20.3.25 `stringer_t* url_decode (stringer_t * s)`

Decode a URL-encoded string into its original representation.

Parameters:

s a managed string containing the UR componentL to be decoded.

Returns:

NULL on failure, or a freshly allocated managed string containing the original data represented by the URL-encoded input on success.

Definition at line 127 of file url.c.

References `hex_decode_chr()`, `hex_valid_chr()`, `log_pedantic`, `st_alloc()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, and `st_length_set()`.

5.20.3.26 stringer_t* url_encode (stringer_t * s)

Encode a data buffer as a valid URL component.

Parameters:

s a managed string containing the data to be encoded.

Returns:

NULL on failure, or a freshly allocated managed string containing the fully-escaped string suitable for use in a URL.

Definition at line 65 of file url.c.

References `HEAP`, `JOINTED`, `log_pedantic`, `MANAGED_T`, `PLACER`, `st_alloc_opts()`, `st_append`, `st_data_get()`, `st_empty_out()`, `st_length_set()`, and `url_valid_chr()`.

5.20.3.27 bool_t url_valid_chr (uchr_t c)

[url.c url.c](#)

Parameters:

c the character to be examined.

Returns:

true if the character is valid in a URL or false if it must be escaped.

Definition at line 15 of file url.c.

Referenced by `url_encode()`, and `url_valid_st()`.

5.20.3.28 size_t url_valid_st (stringer_t * s)

Check a URL string for validity.

Note:

This function confirms that the URL consists of only legal characters and that all escaped sequences are also valid.

Parameters:

s a managed string containing the URL to be verified.

Returns:

0 on failure, or the number of valid characters in the URL if the string is valid.

Definition at line 28 of file url.c.

References `hex_valid_chr()`, `st_empty_out()`, and `url_valid_chr()`.

5.20.3.29 `stringer_t* zbase32_decode (stringer_t * s)`

[zbase32.c](#) [zbase32.c](#)

Parameters:

`s` a managed string containing the data to be decoded.

Returns:

NULL on failure, or a freshly allocated managed string containing the zbase32-decoded data on success.

Definition at line 64 of file `zbase32.c`.

References `log_pedantic`, `mappings`, `st_alloc()`, `st_data_get()`, `st_empty_out()`, `st_free()`, `st_length_set()`, `mappings_t::values`, and `mappings_t::zbase32`.

Referenced by `sess_get()`.

5.20.3.30 `stringer_t* zbase32_encode (stringer_t * s)`

Encode data as a zbase32 string.

Parameters:

`s` a managed string containing the data to be encoded.

Returns:

NULL on failure, or a freshly allocated managed string containing the zbase32-encoded data on success.

Definition at line 16 of file `zbase32.c`.

References `mappings_t::characters`, `log_pedantic`, `mappings`, `st_alloc()`, `st_data_get()`, `st_empty_out()`, `st_length_set()`, and `mappings_t::zbase32`.

Referenced by `sess_token()`.

5.20.4 Variable Documentation

5.20.4.1 `mappings_t mappings`

Definition at line 10 of file `mappings.c`.

Referenced by `base64_decode()`, `base64_decode_mod()`, `base64_encode()`, `base64_encode_mod()`, `base64_encode_wrap()`, `zbase32_decode()`, and `zbase32_encode()`.

5.21 src/core/encodings/hex.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t hex_valid_chr (uchar_t c)`
Determine whether a character is a valid hexadecimal (base 16) character.
- `size_t hex_valid_st (stringer_t *s)`
Determine whether a managed string is a properly formatted hex string and return the number found.
- `size_t hex_count_st (stringer_t *s)`
Count the number of hex characters in a string.
- `uchar_t * hex_encode_chr (byte_t b, uchar_t *output)`
- `stringer_t * hex_encode_st (stringer_t *b, stringer_t *output)`
Convert a block of binary data into a hex string.
- `stringer_t * hex_encode_st_debug (stringer_t *input, size_t maxlen)`
Encode data in a human readable way, for debugging purposes.
- `byte_t hex_decode_chr (uchar_t a, uchar_t b)`
Decode a hex character pair as a single byte (usually for URL decoding).
- `stringer_t * hex_decode_st (stringer_t *h, stringer_t *output)`
Convert a hex string into a binary data blob.
- `stringer_t * hex_encode_opts (stringer_t *input, uint32_t opts)`
Allocates an output string of appropriate size with specified opts for hex encoding of input.
- `stringer_t * hex_decode_opts (stringer_t *input, uint32_t opts)`
Allocates an output string of appropriate size with specified opts for hex decoding of input.

5.21.1 Function Documentation

5.21.1.1 `size_t hex_count_st (stringer_t * s)`

Count the number of hex characters in a string.

Parameters:

`s` the managed string to be scanned.

Returns:

the number of valid hexadecimal characters found in the string.

Definition at line 58 of file hex.c.

References `hex_valid_chr()`, and `st_empty_out()`.

Referenced by `hex_decode_st()`.

5.21.1.2 `byte_t hex_decode_chr (uchr_t a, uchr_t b)`

Decode a hex character pair as a single byte (usually for URL decoding).

Parameters:

- a* the higher order 4-bit hexadecimal character of the byte to be encoded.
- b* the lower order 4-bit hexadecimal character of the byte to be decoded.

Returns:

- a byte containing the value of the decoded hexadecimal character pair, or 0 on failure.

Definition at line 252 of file hex.c.

References `hex_valid_chr()`, `log_pedantic`, and `lower_chr()`.

Referenced by `hex_decode_st()`, `http_data_value_decode()`, `qp_decode()`, and `url_decode()`.

5.21.1.3 `stringer_t* hex_decode_opts (stringer_t * input, uint32_t opts)`

Allocates an output string of appropriate size with specified opts for hex decoding of input.

Parameters:

- input* Input stringer to be decoded.

Returns:

- NULL on failure, otherwise allocated stringer with decoded data.

Definition at line 383 of file hex.c.

References `hex_decode_st()`, `log_error`, `log_pedantic`, `st_alloc_opts()`, `st_empty`, `st_free()`, and `st_length_get()`.

5.21.1.4 `stringer_t* hex_decode_st (stringer_t * h, stringer_t * output)`

Convert a hex string into a binary data blob.

Note:

All hex strings should be composed of pairs of two hex characters representing individual bytes. Invalid hex characters will simply be ignored during processing.

Parameters:

- h* a managed string containing the input hex string to be decoded.
- output* if not NULL, a pointer to a managed string to contain the decoded binary output; if NULL, a new string will be allocated and returned to the caller.

Returns:

- a pointer to a managed string containing the decoded output, or NULL on failure.

Definition at line 287 of file hex.c.

References `hex_count_st()`, `hex_decode_chr()`, `hex_valid_chr()`, `log_pedantic`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `auth_data_fetch()`, and `hex_decode_opts()`.

5.21.1.5 `uchar_t* hex_encode_chr (byte_t b, uchar_t * output)`

Converts a binary octet into a pair of lowercase hex characters. The result is returned inside a thread specific static character buffer. If a valid pointer is also supplied using the output variable then the result will also be written to the memory the pointer indicates.

Parameters:

b The binary octet being encoded.

output A pointer to the memory where the result should be stored or NULL if the result should only be returned.

Returns:

Returns the two character hex representation of the binary input using a thread localized character buffer.

Definition at line 84 of file hex.c.

References `mm_wipe()`, and `number`.

Referenced by `hex_encode_st()`, and `hex_encode_st_debug()`.

5.21.1.6 `stringer_t* hex_encode_opts (stringer_t * input, uint32_t opts)`

Allocates an output string of appropriate size with specified opts for hex encoding of input.

Parameters:

input Input stringer to be encoded.

Returns:

NULL on failure, otherwise allocated stringer with encoded data.

Definition at line 346 of file hex.c.

References `hex_encode_st()`, `log_error`, `log_pedantic`, `st_alloc_opts()`, `st_empty`, `st_free()`, and `st_length_get()`.

5.21.1.7 `stringer_t* hex_encode_st (stringer_t * b, stringer_t * output)`

Convert a block of binary data into a hex string.

Parameters:

b a managed string containing the raw data to be encoded.

output if not NULL, a pointer to a managed string that will store the encoded output; if NULL, a new managed string will be allocated and returned to the caller.

Returns:

NULL on failure, or a pointer to a managed string containing the hex-encoded output on success.

Definition at line 124 of file hex.c.

References `hex_encode_chr()`, `log_pedantic`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `auth_login()`, and `hex_encode_opts()`.

5.21.1.8 `stringer_t* hex_encode_st_debug(stringer_t *input, size_t maxlen)`

Encode data in a human readable way, for debugging purposes.

Note:

All printable ASCII data will be displayed as-is; all other characters will be shown as formatted hex pairs.

Parameters:

input a pointer to a managed string containing the data to be formatted.

maxlen the maximum number of bytes to be displayed. If more characters are available, an ellipsis will be shown in the middle of the string to represent the abridged data, with only the leading and trailing characters returned as part of the display string.

Returns:

NULL on failure, or a pointer to a newly allocated managed string containing the encoded debug data on success.

Definition at line 178 of file hex.c.

References `chr_printable()`, `hex_encode_chr()`, `log_error`, `st_alloc()`, `st_char_get()`, `st_length_get()`, and `st_length_set()`.

5.21.1.9 `bool_t hex_valid_chr(uchar_t c)`

Determine whether a character is a valid hexadecimal (base 16) character. [hex.c](#)

Parameters:

c the character to be tested.

Returns:

true if the character is a valid hexadecimal character; false otherwise.

Definition at line 15 of file hex.c.

Referenced by `hex_count_st()`, `hex_decode_chr()`, `hex_decode_st()`, `hex_valid_st()`, `qp_decode()`, `url_decode()`, and `url_valid_st()`.

5.21.1.10 `size_t hex_valid_st(stringer_t *s)`

Determine whether a managed string is a properly formatted hex string and return the number found.

Note:

A valid hex string consists of only hexadecimal characters and whitespace, and the number of hex characters is divisible by two.

Parameters:

s the managed string to be tested.

Returns:

0 on failure, or the number of hexadecimal characters found in the managed string.

Definition at line 28 of file hex.c.

References `hex_valid_chr()`, and `st_empty_out()`.

5.22 src/core/encodings/mappings.c File Reference

```
#include "magma.h"
```

Variables

- [mappings_t mappings](#)

5.22.1 Variable Documentation

5.22.1.1 mappings_t mappings

Initial value:

```
{
    .zbase32 = {
        .characters = "ybndrfg8ejkmcpqxotluwisa345h769",
        .values = {
            255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
            255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
            255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
            255, 255, 255, 255, 255, 255, 255, 255, 18, 255, 25, 26,
            27, 30, 29, 7, 31, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 2
            55, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
            255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
            255, 24, 1, 12, 3, 8, 5, 6, 28, 21, 9, 10, 255, 11, 2, 16,
            13, 14, 4, 22, 17, 19, 255, 20, 15, 0, 23, 255, 255, 255, 255, 255
        }
    },
    .base64 = {
        .characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
            +/",
        .values = {
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 62,
            0, 0, 0, 63, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 0, 0, 0, 0, 0, 0,
            1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
            17, 18, 19, 20, 21, 22, 23, 24, 25, 0, 0, 0, 0, 0, 26, 27, 28, 29, 30, 3
            1, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
            46, 47, 48, 49, 50, 51, 0, 0, 0, 0, 0
        }
    },
    .base64_mod = {
        .characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
            -_",
        .values = {
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 62, 0, 0, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 0, 0, 0, 0, 0, 0,
            1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
            17, 18, 19, 20, 21, 22, 23, 24, 25, 0, 0, 0, 0, 63, 0, 26, 27, 28, 29, 30,
            31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
            45, 46, 47, 48, 49, 50, 51, 0, 0, 0, 0, 0
        }
    }
}
```

Definition at line 10 of file mappings.c.

Referenced by `base64_decode()`, `base64_decode_mod()`, `base64_encode()`, `base64_encode_mod()`, `base64_encode_wrap()`, `zbase32_decode()`, and `zbase32_encode()`.

5.23 src/core/encodings/qp.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * qp_encode \(stringer_t *s\)](#)
Perform QP (quoted-printable) encoding of a string.
- [stringer_t * qp_decode \(stringer_t *s\)](#)
Perform QP (quoted-printable) decoding of a string.

5.23.1 Function Documentation

5.23.1.1 stringer_t* qp_decode (stringer_t * s)

Perform QP (quoted-printable) decoding of a string. [qp.c](#)

Parameters:

s the managed string containing data to be decoded.

Returns:

a pointer to a managed string containing the 8-bit decoded output, or NULL on failure.

Definition at line 99 of file qp.c.

References [hex_decode_chr\(\)](#), [hex_valid_chr\(\)](#), [log_pedantic](#), [st_alloc\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_length_get\(\)](#), and [st_length_set\(\)](#).

5.23.1.2 stringer_t* qp_encode (stringer_t * s)

Perform QP (quoted-printable) encoding of a string.

Parameters:

s a pointer to a managed string containing data to be encoded.

Returns:

a pointer to a managed string containing the QP encoded data, or NULL on failure.

Definition at line 17 of file qp.c.

References [HEAP](#), [JOINTED](#), [log_pedantic](#), [MANAGED_T](#), [PLACER](#), [QP_LINE_WRAP_LENGTH](#), [st_alloc_opts\(\)](#), [st_append](#), [st_data_get\(\)](#), and [st_empty_out\(\)](#).

Referenced by [mail_build_signature\(\)](#), and [mail_mime_encode_part\(\)](#).

5.24 src/core/encodings/url.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t url_valid_chr (uchar_t c)`
Determine whether a given character is a valid character in a URL.
- `size_t url_valid_st (stringer_t *s)`
Check a URL string for validity.
- `stringer_t * url_encode (stringer_t *s)`
Encode a data buffer as a valid URL component.
- `stringer_t * url_decode (stringer_t *s)`
Decode a URL-encoded string into its original representation.

5.24.1 Function Documentation

5.24.1.1 stringer_t* url_decode (stringer_t * s)

Decode a URL-encoded string into its original representation.

Parameters:

s a managed string containing the UR componentL to be decoded.

Returns:

NULL on failure, or a freshly allocated managed string containing the original data represented by the URL-encoded input on success.

Definition at line 127 of file url.c.

References `hex_decode_chr()`, `hex_valid_chr()`, `log_pedantic`, `st_alloc()`, `st_data_get()`, `st_empty_out()`, `st_length_get()`, and `st_length_set()`.

5.24.1.2 stringer_t* url_encode (stringer_t * s)

Encode a data buffer as a valid URL component.

Parameters:

s a managed string containing the data to be encoded.

Returns:

NULL on failure, or a freshly allocated managed string containing the fully-escaped string suitable for use in a URL.

Definition at line 65 of file url.c.

References `HEAP`, `JOINTED`, `log_pedantic`, `MANAGED_T`, `PLACER`, `st_alloc_opts()`, `st_append`, `st_data_get()`, `st_empty_out()`, `st_length_set()`, and `url_valid_chr()`.

5.24.1.3 `bool_t url_valid_chr (uchar_t c)`

Determine whether a given character is a valid character in a URL. [url.c](#)

Parameters:

`c` the character to be examined.

Returns:

true if the character is valid in a URL or false if it must be escaped.

Definition at line 15 of file `url.c`.

Referenced by `url_encode()`, and `url_valid_st()`.

5.24.1.4 `size_t url_valid_st (stringer_t * s)`

Check a URL string for validity.

Note:

This function confirms that the URL consists of only legal characters and that all escaped sequences are also valid.

Parameters:

`s` a managed string containing the URL to be verified.

Returns:

0 on failure, or the number of valid characters in the URL if the string is valid.

Definition at line 28 of file `url.c`.

References `hex_valid_chr()`, `st_empty_out()`, and `url_valid_chr()`.

5.25 src/core/encodings/zbase32.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * zbase32_encode \(stringer_t *s\)](#)
Encode data as a zbase32 string.
- [stringer_t * zbase32_decode \(stringer_t *s\)](#)
Decode a zbase32 string.

5.25.1 Function Documentation

5.25.1.1 stringer_t* zbase32_decode (stringer_t * s)

Decode a zbase32 string. [zbase32.c](#)

Parameters:

s a managed string containing the data to be decoded.

Returns:

NULL on failure, or a freshly allocated managed string containing the zbase32-decoded data on success.

Definition at line 64 of file [zbase32.c](#).

References [log_pedantic](#), [mappings](#), [st_alloc\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), [st_length_set\(\)](#), [mappings_t::values](#), and [mappings_t::zbase32](#).

Referenced by [sess_get\(\)](#).

5.25.1.2 stringer_t* zbase32_encode (stringer_t * s)

Encode data as a zbase32 string.

Parameters:

s a managed string containing the data to be encoded.

Returns:

NULL on failure, or a freshly allocated managed string containing the zbase32-encoded data on success.

Definition at line 16 of file [zbase32.c](#).

References [mappings_t::characters](#), [log_pedantic](#), [mappings](#), [st_alloc\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_length_set\(\)](#), and [mappings_t::zbase32](#).

Referenced by [sess_token\(\)](#).

5.26 src/core/host/backtrace.c File Reference

A simple function for dumping a stack trace to standard output. `#include "magma.h"`

Functions

- [int_t backtrace_print](#) (void)

Print the current stack backtrace to stdout.

5.26.1 Detailed Description

A simple function for dumping a stack trace to standard output.

Definition in file [backtrace.c](#).

5.26.2 Function Documentation

5.26.2.1 int_t backtrace_print (void)

Print the current stack backtrace to stdout. [backtrace.c](#)

Note:

This function was created because `backtrace_symbols()` can fail due to heap corruption.

Returns:

-1 if the backtrace failed, or 0 on success.

Definition at line 15 of file `backtrace.c`.

5.27 src/core/host/color.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t color_supported](#) (void)
- [const chr_t * color_reset](#) (void)
- [const chr_t * color_red](#) (void)
- [const chr_t * color_green](#) (void)
- [const chr_t * color_yellow](#) (void)
- [const chr_t * color_blue](#) (void)

color.c

- [const chr_t * color_purple](#) (void)
- [const chr_t * color_cyan](#) (void)
- [const chr_t * color_white](#) (void)
- [const chr_t * color_red_bold](#) (void)
- [const chr_t * color_green_bold](#) (void)
- [const chr_t * color_yellow_bold](#) (void)
- [const chr_t * color_blue_bold](#) (void)
- [const chr_t * color_purple_bold](#) (void)
- [const chr_t * color_cyan_bold](#) (void)
- [const chr_t * color_white_bold](#) (void)
- [const chr_t * color_red_underline](#) (void)
- [const chr_t * color_green_underline](#) (void)
- [const chr_t * color_yellow_underline](#) (void)
- [const chr_t * color_blue_underline](#) (void)
- [const chr_t * color_purple_underline](#) (void)
- [const chr_t * color_cyan_underline](#) (void)
- [const chr_t * color_white_underline](#) (void)
- [const chr_t * color_red_intense](#) (void)
- [const chr_t * color_green_intense](#) (void)
- [const chr_t * color_yellow_intense](#) (void)
- [const chr_t * color_blue_intense](#) (void)
- [const chr_t * color_purple_intense](#) (void)
- [const chr_t * color_cyan_intense](#) (void)
- [const chr_t * color_white_intense](#) (void)
- [const chr_t * color_red_intense_bold](#) (void)
- [const chr_t * color_green_intense_bold](#) (void)
- [const chr_t * color_yellow_intense_bold](#) (void)
- [const chr_t * color_blue_intense_bold](#) (void)
- [const chr_t * color_purple_intense_bold](#) (void)
- [const chr_t * color_cyan_intense_bold](#) (void)
- [const chr_t * color_white_intense_bold](#) (void)

5.27.1 Function Documentation

5.27.1.1 [const chr_t* color_blue](#) (void)

[color.c](#)

Definition at line 44 of file [color.c](#).

References [color_supported\(\)](#), and [MAGMA_COLOR_BLUE](#).

5.27.1.2 const chr_t* color_blue_bold (void)

Definition at line 72 of file color.c.

References color_supported(), and MAGMA_COLOR_BLUE_BOLD.

5.27.1.3 const chr_t* color_blue_intense (void)

Definition at line 128 of file color.c.

References color_supported(), and MAGMA_COLOR_BLUE_INTENSE.

5.27.1.4 const chr_t* color_blue_intense_bold (void)

Definition at line 156 of file color.c.

References color_supported(), and MAGMA_COLOR_BLUE_INTENSE_BOLD.

5.27.1.5 const chr_t* color_blue_underline (void)

Definition at line 100 of file color.c.

References color_supported(), and MAGMA_COLOR_BLUE_UNDERLINE.

5.27.1.6 const chr_t* color_cyan (void)

Definition at line 52 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN.

5.27.1.7 const chr_t* color_cyan_bold (void)

Definition at line 80 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN_BOLD.

5.27.1.8 const chr_t* color_cyan_intense (void)

Definition at line 136 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN_INTENSE.

5.27.1.9 const chr_t* color_cyan_intense_bold (void)

Definition at line 164 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN_INTENSE_BOLD.

5.27.1.10 const chr_t* color_cyan_underline (void)

Definition at line 108 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN_UNDERLINE.

5.27.1.11 const chr_t* color_green (void)

Definition at line 36 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN.

5.27.1.12 const chr_t* color_green_bold (void)

Definition at line 64 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN_BOLD.

5.27.1.13 const chr_t* color_green_intense (void)

Definition at line 120 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN_INTENSE.

5.27.1.14 const chr_t* color_green_intense_bold (void)

Definition at line 148 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN_INTENSE_BOLD.

5.27.1.15 const chr_t* color_green_underline (void)

Definition at line 92 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN_UNDERLINE.

5.27.1.16 const chr_t* color_purple (void)

Definition at line 48 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE.

5.27.1.17 const chr_t* color_purple_bold (void)

Definition at line 76 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE_BOLD.

5.27.1.18 const chr_t* color_purple_intense (void)

Definition at line 132 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE_INTENSE.

5.27.1.19 const chr_t* color_purple_intense_bold (void)

Definition at line 160 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE_INTENSE_BOLD.

5.27.1.20 const chr_t* color_purple_underline (void)

Definition at line 104 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE_UNDERLINE.

5.27.1.21 const chr_t* color_red (void)

Definition at line 32 of file color.c.

References color_supported(), and MAGMA_COLOR_RED.

5.27.1.22 const chr_t* color_red_bold (void)

Definition at line 60 of file color.c.

References color_supported(), and MAGMA_COLOR_RED_BOLD.

5.27.1.23 const chr_t* color_red_intense (void)

Definition at line 116 of file color.c.

References color_supported(), and MAGMA_COLOR_RED_INTENSE.

5.27.1.24 const chr_t* color_red_intense_bold (void)

Definition at line 144 of file color.c.

References color_supported(), and MAGMA_COLOR_RED_INTENSE_BOLD.

5.27.1.25 const chr_t* color_red_underline (void)

Definition at line 88 of file color.c.

References color_supported(), and MAGMA_COLOR_RED_UNDERLINE.

5.27.1.26 const chr_t* color_reset (void)

Definition at line 28 of file color.c.

References color_supported(), and MAGMA_COLOR_RESET.

5.27.1.27 bool_t color_supported (void)

Definition at line 10 of file color.c.

References magma_t::daemonize, magma, NULLER, st_cmp_ci_eq(), and magma_t::system.

Referenced by color_blue(), color_blue_bold(), color_blue_intense(), color_blue_intense_bold(), color_blue_underline(), color_cyan(), color_cyan_bold(), color_cyan_intense(), color_cyan_intense_bold(), color_cyan_underline(), color_green(), color_green_bold(), color_green_intense(), color_green_intense_bold(), color_green_underline(), color_purple(), color_purple_bold(), color_purple_intense(), color_purple_intense_bold(), color_purple_underline(), color_red(), color_red_bold(), color_red_intense(), color_red_intense_bold(), color_red_underline(), color_reset(), color_white(), color_white_bold(), color_white_intense(), color_white_intense_bold(), color_white_underline(), color_yellow(), color_yellow_bold(), color_yellow_intense(), color_yellow_intense_bold(), and color_yellow_underline().

5.27.1.28 const chr_t* color_white (void)

Definition at line 56 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE.

5.27.1.29 const chr_t* color_white_bold (void)

Definition at line 84 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE_BOLD.

5.27.1.30 const chr_t* color_white_intense (void)

Definition at line 140 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE_INTENSE.

5.27.1.31 const chr_t* color_white_intense_bold (void)

Definition at line 168 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE_INTENSE_BOLD.

5.27.1.32 const chr_t* color_white_underline (void)

Definition at line 112 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE_UNDERLINE.

5.27.1.33 const chr_t* color_yellow (void)

Definition at line 40 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW.

5.27.1.34 const chr_t* color_yellow_bold (void)

Definition at line 68 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW_BOLD.

5.27.1.35 const chr_t* color_yellow_intense (void)

Definition at line 124 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW_INTENSE.

5.27.1.36 const chr_t* color_yellow_intense_bold (void)

Definition at line 152 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW_INTENSE_BOLD.

5.27.1.37 `const chr_t* color_yellow_underline` (void)

Definition at line 96 of file color.c.

References `color_supported()`, and `MAGMA_COLOR_YELLOW_UNDERLINE`.

5.28 src/core/host/errors.c File Reference

Functions to help handle errno. `#include "magma.h"`

Defines

- `#define MAGMA_EPERM` 1
- `#define MAGMA_ENOENT` 2
- `#define MAGMA_ESRCH` 3
- `#define MAGMA_EINTR` 4
- `#define MAGMA_EIO` 5
- `#define MAGMA_ENXIO` 6
- `#define MAGMA_E2BIG` 7
- `#define MAGMA_ENOEXEC` 8
- `#define MAGMA_EBADF` 9
- `#define MAGMA_ECHILD` 10
- `#define MAGMA_EAGAIN` 11
- `#define MAGMA_ENOMEM` 12
- `#define MAGMA_EACCES` 13
- `#define MAGMA_EFAULT` 14
- `#define MAGMA_ENOTBLK` 15
- `#define MAGMA_EBUSY` 16
- `#define MAGMA_EEXIST` 17
- `#define MAGMA_EXDEV` 18
- `#define MAGMA_ENODEV` 19
- `#define MAGMA_ENOTDIR` 20
- `#define MAGMA_EISDIR` 21
- `#define MAGMA_EINVAL` 22
- `#define MAGMA_ENFILE` 23
- `#define MAGMA_EMFILE` 24
- `#define MAGMA_ENOTTY` 25
- `#define MAGMA_ETXTBSY` 26
- `#define MAGMA_EFBIG` 27
- `#define MAGMA_ENOSPC` 28
- `#define MAGMA_ESPIPE` 29
- `#define MAGMA_EROFS` 30
- `#define MAGMA_EMLINK` 31
- `#define MAGMA_EPIPE` 32
- `#define MAGMA_EDOM` 33
- `#define MAGMA_ERANGE` 34

Functions

- `chr_t * errno_name` (int error)

errors.c

5.28.1 Detailed Description

Functions to help handle errno.

Definition in file [errors.c](#).

5.28.2 Define Documentation

5.28.2.1 **#define MAGMA_E2BIG 7**

Definition at line 43 of file errors.c.

Referenced by `errno_name()`.

5.28.2.2 **#define MAGMA_EACCES 13**

Definition at line 73 of file errors.c.

Referenced by `errno_name()`.

5.28.2.3 **#define MAGMA_EAGAIN 11**

Definition at line 63 of file errors.c.

Referenced by `errno_name()`.

5.28.2.4 **#define MAGMA_EBADF 9**

Definition at line 53 of file errors.c.

Referenced by `errno_name()`.

5.28.2.5 **#define MAGMA_EBUSY 16**

Definition at line 88 of file errors.c.

Referenced by `errno_name()`.

5.28.2.6 **#define MAGMA_ECHILD 10**

Definition at line 58 of file errors.c.

Referenced by `errno_name()`.

5.28.2.7 **#define MAGMA_EDOM 33**

Definition at line 173 of file errors.c.

5.28.2.8 **#define MAGMA_EEXIST 17**

Definition at line 93 of file errors.c.

Referenced by `errno_name()`.

5.28.2.9 **#define MAGMA_EFAULT 14**

Definition at line 78 of file errors.c.

Referenced by `errno_name()`.

5.28.2.10 #define MAGMA_EFBIG 27

Definition at line 143 of file errors.c.

Referenced by `errno_name()`.

5.28.2.11 #define MAGMA_EINTR 4

Definition at line 28 of file errors.c.

Referenced by `errno_name()`.

5.28.2.12 #define MAGMA_EINVAL 22

Definition at line 118 of file errors.c.

Referenced by `errno_name()`.

5.28.2.13 #define MAGMA_EIO 5

Definition at line 33 of file errors.c.

Referenced by `errno_name()`.

5.28.2.14 #define MAGMA_EISDIR 21

Definition at line 113 of file errors.c.

Referenced by `errno_name()`.

5.28.2.15 #define MAGMA_EMFILE 24

Definition at line 128 of file errors.c.

Referenced by `errno_name()`.

5.28.2.16 #define MAGMA_EMLINK 31

Definition at line 163 of file errors.c.

Referenced by `errno_name()`.

5.28.2.17 #define MAGMA_ENFILE 23

Definition at line 123 of file errors.c.

Referenced by `errno_name()`.

5.28.2.18 #define MAGMA_ENODEV 19

Definition at line 103 of file errors.c.

Referenced by `errno_name()`.

5.28.2.19 #define MAGMA_ENOENT 2

Definition at line 18 of file errors.c.

Referenced by `errno_name()`.

5.28.2.20 #define MAGMA_ENOEXEC 8

Definition at line 48 of file errors.c.

Referenced by `errno_name()`.

5.28.2.21 #define MAGMA_ENOMEM 12

Definition at line 68 of file errors.c.

Referenced by `errno_name()`.

5.28.2.22 #define MAGMA_ENOSPC 28

Definition at line 148 of file errors.c.

Referenced by `errno_name()`.

5.28.2.23 #define MAGMA_ENOTBLK 15

Definition at line 83 of file errors.c.

Referenced by `errno_name()`.

5.28.2.24 #define MAGMA_ENOTDIR 20

Definition at line 108 of file errors.c.

Referenced by `errno_name()`.

5.28.2.25 #define MAGMA_ENOTTY 25

Definition at line 133 of file errors.c.

Referenced by `errno_name()`.

5.28.2.26 #define MAGMA_ENXIO 6

Definition at line 38 of file errors.c.

Referenced by `errno_name()`.

5.28.2.27 #define MAGMA_EPERM 1

Definition at line 13 of file errors.c.

Referenced by `errno_name()`.

5.28.2.28 #define MAGMA_EPIPE 32

Definition at line 168 of file errors.c.

Referenced by `errno_name()`.

5.28.2.29 #define MAGMA_ERANGE 34

Definition at line 178 of file errors.c.

5.28.2.30 #define MAGMA_EROFS 30

Definition at line 158 of file errors.c.

Referenced by `errno_name()`.

5.28.2.31 #define MAGMA_ESPIPE 29

Definition at line 153 of file errors.c.

Referenced by `errno_name()`.

5.28.2.32 #define MAGMA_ESRCH 3

Definition at line 23 of file errors.c.

Referenced by `errno_name()`.

5.28.2.33 #define MAGMA_ETXTBSY 26

Definition at line 138 of file errors.c.

Referenced by `errno_name()`.

5.28.2.34 #define MAGMA_EXDEV 18

Definition at line 98 of file errors.c.

Referenced by `errno_name()`.

5.28.3 Function Documentation**5.28.3.1 `chr_t* errno_name (int error)`**

errors.c

Definition at line 181 of file errors.c.

References `MAGMA_E2BIG`, `MAGMA_EACCES`, `MAGMA_EAGAIN`, `MAGMA_EBADF`, `MAGMA_EBUSY`, `MAGMA_ECHILD`, `MAGMA_EEXIST`, `MAGMA_EFAULT`, `MAGMA_EFBIG`, `MAGMA_EINTR`, `MAGMA_EINVAL`, `MAGMA_EIO`, `MAGMA_EISDIR`, `MAGMA_EMFILE`, `MAGMA_EMLINK`, `MAGMA_ENFILE`, `MAGMA_ENODEV`, `MAGMA_ENOENT`, `MAGMA_ENOEXEC`, `MAGMA_ENOMEM`, `MAGMA_ENOSPC`, `MAGMA_ENOTBLK`, `MAGMA_ENOTDIR`, `MAGMA_ENOTTY`, `MAGMA_ENXIO`, `MAGMA_EPERM`, `MAGMA_EPIPE`, `MAGMA_EROFS`, `MAGMA_ESPIPE`, `MAGMA_ESRCH`, `MAGMA_ETXTBSY`, and `MAGMA_EXDEV`.

Referenced by `tcp_continue()`, `tls_continue()`, and `tls_server_alloc()`.

5.29 src/servers/http/errors.c File Reference

```
#include "magma.h"
```

Functions

- void `http_print_301` (`connection_t` *con, `chr_t` *location, `int_t` tls)
Redirect the browser to a new location with an HTTP 301 response.
- void `http_print_400` (`connection_t` *con)
Return an HTTP 400 bad client request response to the client.
- void `http_print_403` (`connection_t` *con)
Return an HTTP 403 access denied response to the client.
- void `http_print_404` (`connection_t` *con)
Return an HTTP 404 location not found response to the client.
- void `http_print_405` (`connection_t` *con)
Return an HTTP 405 method not allowed response to the client.
- void `http_print_500` (`connection_t` *con)
Return an HTTP 500 internal server error response to the client.
- void `http_print_500_log` (`connection_t` *con, `chr_t` *logmsg)
Return an HTTP 500 internal server error response to the client, with additional logging information.
- void `http_print_501` (`connection_t` *con)
Return an HTTP 501 method not implemented response to the client.

5.29.1 Function Documentation

5.29.1.1 void http_print_301 (connection_t * con, chr_t * location, int_t tls)

Redirect the browser to a new location with an HTTP 301 response. errors.c

Note:

SSL downgrades are not possible; in order for an ssl upgrade, magma.web.ssl_redirect must first be set.

Parameters:

con the client's http connection handle.
location the url to which the client will be redirected.
tls if 1, direct to `https://` else direct to `http://` address.

Returns:

This function returns no value.

LOW: This function should probably move to the response file and be updated to use the cookie/xss helper functions.

Definition at line 18 of file errors.c.

References `con_print()`, `con_secure()`, `HTTP_COMPLETE`, `http_print_403()`, `http_print_500()`, `log_pedantic`, `magma`, `st_char_get()`, `st_cleanup`, `st_duped()`, `st_free()`, `st_length_int()`, `st_merge`, `magma_t::tls_redirect`, and `magma_t::web`.

Referenced by `contact_process()`, `portal_endpoint()`, `portal_process()`, `portal_upload()`, `register_process()`, and `teacher_process()`.

5.29.1.2 void http_print_400 (connection_t * con)

Return an HTTP 400 bad client request response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 80 of file `errors.c`.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_requeue()`.

5.29.1.3 void http_print_403 (connection_t * con)

Return an HTTP 403 access denied response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 94 of file `errors.c`.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_print_301()`, and `http_requeue()`.

5.29.1.4 void http_print_404 (connection_t * con)

Return an HTTP 404 location not found response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 108 of file `errors.c`.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_requeue()`.

5.29.1.5 void http_print_405 (connection_t * con)

Return an HTTP 405 method not allowed response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 122 of file errors.c.

References con_write_st(), HTTP_CONNECTION_CLOSE, http_response_header(), and PLACER.

Referenced by http_requeue().

5.29.1.6 void http_print_500 (connection_t * con)

Return an HTTP 500 internal server error response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 136 of file errors.c.

References con_write_st(), HTTP_CONNECTION_CLOSE, http_response_header(), log_options, M_LOG_CRITICAL, M_LOG_STACK_TRACE, and PLACER.

Referenced by contact_print_message(), http_print_301(), http_requeue(), portal_print_login(), register_print_captcha(), register_print_message(), register_print_step3(), statistics_process(), teacher_print_form(), and teacher_print_message().

5.29.1.7 void http_print_500_log (connection_t * con, chr_t * logmsg)

Return an HTTP 500 internal server error response to the client, with additional logging information.

Parameters:

con the client's http connection handle.

logmsg a pointer to a null-terminated string with an message describing the cause of the http 500 error.

Returns:

This function returns no value.

Definition at line 154 of file errors.c.

References con_write_st(), HTTP_CONNECTION_CLOSE, http_response_header(), log_options, M_LOG_CRITICAL, and PLACER.

Referenced by contact_print_message(), register_print_step1(), and register_print_step2().

5.29.1.8 void http_print_501 (connection_t * con)

Return an HTTP 501 method not implemented response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 169 of file errors.c.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_requeue()`.

5.30 src/core/host/files.c File Reference

```
#include "magma.h"
```

Functions

- [int_t file_read](#) (const char *name, [stringer_t](#) *output)
Get the contents of a file on disk.
- [stringer_t * file_load](#) (const char *name)
Get the contents of a file on disk.
- [int_t file_temp_handle](#) ([chr_t](#) *pdir, [stringer_t](#) **tmpname)
Get a file handle to a temporary file created in a specified directory.
- [bool_t file_accessible](#) (const [chr_t](#) *path)
Determine whether a given filename or directory path is accessible by a user.
- [bool_t file_readwritable](#) (const [chr_t](#) *path)
Determine whether a given filename or directory path is readable and writable by a user.
- [bool_t file_world_accessible](#) (const [chr_t](#) *path)
Determine whether a given filename or directory path is readable or writable by other users.

5.30.1 Function Documentation

5.30.1.1 [bool_t file_accessible](#) (const [chr_t](#) * *path*)

Determine whether a given filename or directory path is accessible by a user.

Parameters:

path a null-terminated string with the name of the filename or directory to be checked for existence/access.

Returns:

true if the pathname exists and is readable, or false otherwise.

Definition at line 155 of file files.c.

5.30.1.2 [stringer_t* file_load](#) (const char * *name*)

Get the contents of a file on disk. [files.c](#)

Parameters:

name a character pointer to the full pathname of the file to be opened.

Returns:

NULL on failure, or a managed string containing all the data in the file.

Definition at line 51 of file files.c.

References [log_info](#), [st_alloc\(\)](#), [st_avail_get\(\)](#), [st_data_get\(\)](#), and [st_length_set\(\)](#).

Referenced by [config_load_file_settings\(\)](#), [dkim_start\(\)](#), and [prime_start\(\)](#).

5.30.1.3 `int_t file_read (const char * name, stringer_t * output)`

Get the contents of a file on disk.

See also:

[file_load\(\)](#)

Note:

This is similar in function to [file_load\(\)](#) but there is no `read()` length checking, so it can be used on non-regular files.

Parameters:

name a character pointer to the full pathname of the file to be opened.

output a managed string where the results of the file read operation will be stored.

Returns:

-1 on failure or the number of bytes read from the file on success.

Definition at line 18 of file files.c.

References `log_info`, `log_pedantic`, `MEMORYBUF`, `st_avail_get()`, `st_data_get()`, and `st_length_set()`.

Referenced by `process_find_pid()`, and `process_kill()`.

5.30.1.4 `bool_t file_readwritable (const chr_t * path)`

Determine whether a given filename or directory path is readable and writable by a user.

Parameters:

path a null-terminated string with the name of the filename or directory to be checked for permissions.

Returns:

true if the pathname exists and is readable and writable, or false otherwise.

Definition at line 165 of file files.c.

Referenced by `servers_validate()`.

5.30.1.5 `int_t file_temp_handle (chr_t * pdir, stringer_t ** tmpname)`

Get a file handle to a temporary file created in a specified directory.

Parameters:

pdir the parent directory in which to create the temporary file, or NULL for the default spool dir.

tmpname an optional pointer to a managed string to receive the name of the created temp file.

Returns:

-1 on failure or the new temporary file's file descriptor on success.

Definition at line 98 of file files.c.

References `CONSTANT`, `HEAP`, `JOINTED`, `log_pedantic`, `MAGMA_SPOOL_BASE`, `MANAGED_T`, `ns_length_get()`, `spool_path()`, `st_append`, `st_char_get()`, `st_dupe()`, `st_dupe_opts()`, `st_free()`, and `st_import()`.

5.30.1.6 bool_t file_world_accessible (const chr_t * *path*)

Determine whether a given filename or directory path is readable or writable by other users.

Parameters:

path a null-terminated string with the name of the filename or directory to be checked for world permissions.

Returns:

true if the pathname exists and is readable or writable by others, or false otherwise.

Definition at line 175 of file files.c.

Referenced by dkim_start(), prime_start(), and servers_validate().

5.31 src/core/host/folder.c File Reference

```
#include "magma.h"
```

Functions

- [int_t folder_exists](#) ([stringer_t](#) *path, [bool_t](#) create)
Check to see if a specified directory exists, or if specified, create it if it doesn't exist.
- [int_t folder_count](#) ([stringer_t](#) *path, [bool_t](#) recursive, [bool_t](#) strict)
Count the number of files in a folder.

5.31.1 Function Documentation

5.31.1.1 int_t folder_count (stringer_t * path, bool_t recursive, bool_t strict)

Count the number of files in a folder. [folder.c](#)

Definition at line 39 of file folder.c.

References [count](#), [folder_count\(\)](#), [log_info](#), [MEMORYBUF](#), [NULLER](#), [PLACER](#), [st_char_get\(\)](#), [st_cmp_cs_ends\(\)](#), [st_cmp_cs_eq\(\)](#), [st_empty](#), [st_free\(\)](#), and [st_merge](#).

Referenced by [folder_count\(\)](#).

5.31.1.2 int_t folder_exists (stringer_t * path, bool_t create)

Check to see if a specified directory exists, or if specified, create it if it doesn't exist.

Parameters:

path a managed string containing the full pathname of the directory.

create if true, attempt to create the directory if it does not already exist.

Returns:

-1 if the directory doesn't exist or couldn't be created, 0 if the directory exists, or 1 if the directory was created.

Definition at line 16 of file folder.c.

References [st_char_get\(\)](#).

Referenced by [log_start\(\)](#), and [spool_check\(\)](#).

5.32 src/core/host/host.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * host_platform](#) ([stringer_t](#) *output)
Get a description of the local operating system.
- [stringer_t * host_version](#) ([stringer_t](#) *output)
Get release information about the local OS.

5.32.1 Function Documentation

5.32.1.1 [stringer_t * host_platform](#) ([stringer_t](#) * *output*)

Get a description of the local operating system. [host.c](#)

Parameters:

output a pointer to a managed string to receive the OS description.

Returns:

NULL on failure, or the user-specified managed string containing the OS info on success.

Definition at line 15 of file [host.c](#).

References [log_pedantic](#), [ns_length_get\(\)](#), [st_avail_get\(\)](#), and [st_sprint\(\)](#).

Referenced by [lib_load\(\)](#).

5.32.1.2 [stringer_t * host_version](#) ([stringer_t](#) * *output*)

Get release information about the local OS.

Parameters:

output a pointer to a managed string to receive the release information.

Returns:

NULL on failure, or the user-specified managed string containing the release info on success.

Definition at line 41 of file [host.c](#).

References [log_pedantic](#), [ns_length_get\(\)](#), [st_avail_get\(\)](#), and [st_sprint\(\)](#).

Referenced by [lib_load\(\)](#).

5.33 src/core/host/host.h File Reference

Data Structures

- struct [ip_t](#)
- struct [subnet_t](#)

Defines

- #define [MAGMA_COLOR_RESET](#) "\033[m"
- #define [MAGMA_COLOR_RED](#) "\033[0;31m"
- #define [MAGMA_COLOR_GREEN](#) "\033[0;32m"
- #define [MAGMA_COLOR_YELLOW](#) "\033[0;33m"
- #define [MAGMA_COLOR_BLUE](#) "\033[0;34m"
- #define [MAGMA_COLOR_PURPLE](#) "\033[0;35m"
- #define [MAGMA_COLOR_CYAN](#) "\033[0;36m"
- #define [MAGMA_COLOR_WHITE](#) "\033[0;37m"
- #define [MAGMA_COLOR_RED_BOLD](#) "\033[1;31m"
- #define [MAGMA_COLOR_GREEN_BOLD](#) "\033[1;32m"
- #define [MAGMA_COLOR_YELLOW_BOLD](#) "\033[1;33m"
- #define [MAGMA_COLOR_BLUE_BOLD](#) "\033[1;34m"
- #define [MAGMA_COLOR_PURPLE_BOLD](#) "\033[1;35m"
- #define [MAGMA_COLOR_CYAN_BOLD](#) "\033[1;36m"
- #define [MAGMA_COLOR_WHITE_BOLD](#) "\033[1;37m"
- #define [MAGMA_COLOR_RED_UNDERLINE](#) "\033[4;31m"
- #define [MAGMA_COLOR_GREEN_UNDERLINE](#) "\033[4;32m"
- #define [MAGMA_COLOR_YELLOW_UNDERLINE](#) "\033[4;33m"
- #define [MAGMA_COLOR_BLUE_UNDERLINE](#) "\033[4;34m"
- #define [MAGMA_COLOR_PURPLE_UNDERLINE](#) "\033[4;35m"
- #define [MAGMA_COLOR_CYAN_UNDERLINE](#) "\033[4;36m"
- #define [MAGMA_COLOR_WHITE_UNDERLINE](#) "\033[4;37m"
- #define [MAGMA_COLOR_RED_INTENSE](#) "\033[0;91m"
- #define [MAGMA_COLOR_GREEN_INTENSE](#) "\033[0;92m"
- #define [MAGMA_COLOR_YELLOW_INTENSE](#) "\033[0;93m"
- #define [MAGMA_COLOR_BLUE_INTENSE](#) "\033[0;94m"
- #define [MAGMA_COLOR_PURPLE_INTENSE](#) "\033[0;95m"
- #define [MAGMA_COLOR_CYAN_INTENSE](#) "\033[0;96m"
- #define [MAGMA_COLOR_WHITE_INTENSE](#) "\033[0;97m"
- #define [MAGMA_COLOR_RED_INTENSE_BOLD](#) "\033[1;91m"
- #define [MAGMA_COLOR_GREEN_INTENSE_BOLD](#) "\033[1;92m"
- #define [MAGMA_COLOR_YELLOW_INTENSE_BOLD](#) "\033[1;93m"
- #define [MAGMA_COLOR_BLUE_INTENSE_BOLD](#) "\033[1;94m"
- #define [MAGMA_COLOR_PURPLE_INTENSE_BOLD](#) "\033[1;95m"
- #define [MAGMA_COLOR_CYAN_INTENSE_BOLD](#) "\033[1;96m"
- #define [MAGMA_COLOR_WHITE_INTENSE_BOLD](#) "\033[1;97m"
- #define [MAGMA_PROC_PATH](#) "/proc"

Typedefs

- typedef int16_t [octet_t](#)
- typedef int32_t [segment_t](#)

Enumerations

- enum { [MAGMA_SPOOL_BASE](#) = 0, [MAGMA_SPOOL_DATA](#) = 1, [MAGMA_SPOOL_SCAN](#) = 2 }

Functions

- [int_t backtrace_print](#) (void)
backtrace.c
- const [chr_t](#) * [color_blue](#) (void)
color.c
- const [chr_t](#) * [color_blue_bold](#) (void)
- const [chr_t](#) * [color_blue_intense](#) (void)
- const [chr_t](#) * [color_blue_intense_bold](#) (void)
- const [chr_t](#) * [color_blue_underline](#) (void)
- const [chr_t](#) * [color_cyan](#) (void)
- const [chr_t](#) * [color_cyan_bold](#) (void)
- const [chr_t](#) * [color_cyan_intense](#) (void)
- const [chr_t](#) * [color_cyan_intense_bold](#) (void)
- const [chr_t](#) * [color_cyan_underline](#) (void)
- const [chr_t](#) * [color_green](#) (void)
- const [chr_t](#) * [color_green_bold](#) (void)
- const [chr_t](#) * [color_green_intense](#) (void)
- const [chr_t](#) * [color_green_intense_bold](#) (void)
- const [chr_t](#) * [color_green_underline](#) (void)
- const [chr_t](#) * [color_purple](#) (void)
- const [chr_t](#) * [color_purple_bold](#) (void)
- const [chr_t](#) * [color_purple_intense](#) (void)
- const [chr_t](#) * [color_purple_intense_bold](#) (void)
- const [chr_t](#) * [color_purple_underline](#) (void)
- const [chr_t](#) * [color_red](#) (void)
- const [chr_t](#) * [color_red_bold](#) (void)
- const [chr_t](#) * [color_red_intense](#) (void)
- const [chr_t](#) * [color_red_intense_bold](#) (void)
- const [chr_t](#) * [color_red_underline](#) (void)
- const [chr_t](#) * [color_reset](#) (void)
- [bool_t color_supported](#) (void)
- const [chr_t](#) * [color_white](#) (void)
- const [chr_t](#) * [color_white_bold](#) (void)
- const [chr_t](#) * [color_white_intense](#) (void)
- const [chr_t](#) * [color_white_intense_bold](#) (void)
- const [chr_t](#) * [color_white_underline](#) (void)
- const [chr_t](#) * [color_yellow](#) (void)
- const [chr_t](#) * [color_yellow_bold](#) (void)
- const [chr_t](#) * [color_yellow_intense](#) (void)
- const [chr_t](#) * [color_yellow_intense_bold](#) (void)
- const [chr_t](#) * [color_yellow_underline](#) (void)
- [stringer_t](#) * [file_load](#) (const char *name)
files.c
- [int_t file_read](#) (const char *name, [stringer_t](#) *output)
Get the contents of a file on disk.

- [int_t file_temp_handle](#) ([chr_t](#) *pdir, [stringer_t](#) **tmpname)
Get a file handle to a temporary file created in a specified directory.
- [bool_t file_accessible](#) (const [chr_t](#) *path)
Determine whether a given filename or directory path is accessible by a user.
- [bool_t file_readwritable](#) (const [chr_t](#) *path)
Determine whether a given filename or directory path is readable and writable by a user.
- [bool_t file_world_accessible](#) (const [chr_t](#) *path)
Determine whether a given filename or directory path is readable or writable by other users.
- [ip_t](#) * [tcp_addr_ip](#) (int sockd, [ip_t](#) *output)
tcp.c
- [stringer_t](#) * [tcp_addr_st](#) (int sockd, [stringer_t](#) *output)
- int [tcp_continue](#) (int sockd, int result, int syserror)
Return -1 if the connection is invalid, 0 if the operation should be retried, or a positive number indicating the number of bytes processed.
- [int_t tcp_error](#) (int error)
Determine whether the error is a permanent/fatal failure, or a transient error.
- int [tcp_read](#) (int sockd, void *buffer, int length, [bool_t](#) block)
Read data from a TCP/IP network socket.
- [int_t tcp_status](#) (int sockd)
Determine whether the socket connection provided by sockd is still valid.
- int [tcp_wait](#) (int sockd)
Blocks until a socket is ready for a read/write operation, or otherwise becomes invalid.
- int [tcp_write](#) (int sockd, const void *buffer, int length, [bool_t](#) block)
Write data to an open TCP/IP network socket.
- [stringer_t](#) * [host_platform](#) ([stringer_t](#) *output)
host.c
- [stringer_t](#) * [host_version](#) ([stringer_t](#) *output)
Get release information about the local OS.
- [chr_t](#) * [errno_name](#) (int error)
errors.c
- [chr_t](#) * [signal_name](#) (int signal, char *buffer, [size_t](#) length)
signals.c
- [int_t folder_count](#) ([stringer_t](#) *path, [bool_t](#) recursive, [bool_t](#) strict)
folder.c
- [int_t folder_exists](#) ([stringer_t](#) *path, [bool_t](#) create)
Check to see if a specified directory exists, or if specified, create it if it doesn't exist.

- `pid_t process_find_pid (stringer_t *name)`
process.c
- `int_t process_kill (stringer_t *name, int_t signum, int_t wait)`
Kill a named process with the specified signal, and retry if necessary.
- `pid_t process_my_pid (void)`
Return the current process identifier using appropriate function for the current system.
- `int_t spool_check (stringer_t *path)`
spool.c
- `int_t spool_check_file (const char *file, const struct stat *info, int type)`
An internal function used by the `ftw()` function to cleanup the file contents of a spool directory.
- `int_t spool_cleanup (void)`
Clean up the file contents in the spool base directory.
- `uint64_t spool_error_stats (void)`
Get the number of spool errors encountered.
- `int_t spool_mktemp (int_t spool, chr_t *prefix)`
Create a temporary file in a specified spool directory.
- `stringer_t * spool_path (int_t spool)`
Get the full path to a requested spool directory.
- `bool_t spool_start (void)`
Create and check the spool directories, and clean them for use.
- `void spool_stop (void)`
Clean up the file contents in the spool base directory.
- `bool_t ip_addr_eq (ip_t *ip1, ip_t *ip2)`
ip.c
- `ip_t * ip_copy (ip_t *dst, ip_t *src)`
Create a copy of an IP address object.
- `int_t ip_family (ip_t *address)`
An abstract method of retrieving the address family for an IP structure.
- `bool_t ip_localhost (ip_t *address)`
Determine whether an IP address matches the localhost loopback address.
- `bool_t ip_matches_subnet (subnet_t *subnet, ip_t *addr)`
Determines whether a given IP address matches a subnet mask.
- `octet_t ip_octet (ip_t *address, int_t position)`
Extract a specified 8 bit octet from an IPv4 or IPv6 address.
- `stringer_t * ip_presentation (ip_t *address, stringer_t *output)`
Convert an IP address structure into a readable string.

- `bool_t ip_private (ip_t *address)`
Determine whether an IP address appears to be associated with a non-routeable, private address space.
- `stringer_t * ip_reversed (ip_t *address, stringer_t *output)`
Get a reversed-IP string, for use in an RBL lookup.
- `segment_t ip_segment (ip_t *address, int_t position)`
Extract a specified 16 bit segment from an IPv4 or IPv6 address.
- `stringer_t * ip_standard (ip_t *address, stringer_t *output)`
Convert an IP address structure to string representation.
- `bool_t ip_addr_st (chr_t *ipstr, ip_t *out)`
Convert a string into an IP address object.
- `bool_t ip_subnet_st (chr_t *substr, subnet_t *out)`
Convert a string to an IP address object.
- `stringer_t * ip_subnet (ip_t *address, stringer_t *output)`
Display the simple subnet string for an IP address.
- `int8_t ip_type (ip_t *address)`
Determine whether a structure holds an IPv4 or IPv6 address.
- `uint32_t ip_word (ip_t *address, int_t position)`
Get a specified 32-bit segment of an IP address.

5.33.1 Define Documentation

5.33.1.1 `#define MAGMA_COLOR_BLUE "\033[0;34m"`

Definition at line 16 of file host.h.

Referenced by `color_blue()`.

5.33.1.2 `#define MAGMA_COLOR_BLUE_BOLD "\033[1;34m"`

Definition at line 24 of file host.h.

Referenced by `color_blue_bold()`.

5.33.1.3 `#define MAGMA_COLOR_BLUE_INTENSE "\033[0;94m"`

Definition at line 40 of file host.h.

Referenced by `color_blue_intense()`.

5.33.1.4 `#define MAGMA_COLOR_BLUE_INTENSE_BOLD "\033[1;94m"`

Definition at line 48 of file host.h.

Referenced by `color_blue_intense_bold()`.

5.33.1.5 #define MAGMA_COLOR_BLUE_UNDERLINE "\033[4;34m"

Definition at line 32 of file host.h.

Referenced by color_blue_underline().

5.33.1.6 #define MAGMA_COLOR_CYAN "\033[0;36m"

Definition at line 18 of file host.h.

Referenced by color_cyan().

5.33.1.7 #define MAGMA_COLOR_CYAN_BOLD "\033[1;36m"

Definition at line 26 of file host.h.

Referenced by color_cyan_bold().

5.33.1.8 #define MAGMA_COLOR_CYAN_INTENSE "\033[0;96m"

Definition at line 42 of file host.h.

Referenced by color_cyan_intense().

5.33.1.9 #define MAGMA_COLOR_CYAN_INTENSE_BOLD "\033[1;96m"

Definition at line 50 of file host.h.

Referenced by color_cyan_intense_bold().

5.33.1.10 #define MAGMA_COLOR_CYAN_UNDERLINE "\033[4;36m"

Definition at line 34 of file host.h.

Referenced by color_cyan_underline().

5.33.1.11 #define MAGMA_COLOR_GREEN "\033[0;32m"

Definition at line 14 of file host.h.

Referenced by color_green().

5.33.1.12 #define MAGMA_COLOR_GREEN_BOLD "\033[1;32m"

Definition at line 22 of file host.h.

Referenced by color_green_bold().

5.33.1.13 #define MAGMA_COLOR_GREEN_INTENSE "\033[0;92m"

Definition at line 38 of file host.h.

Referenced by color_green_intense().

5.33.1.14 #define MAGMA_COLOR_GREEN_INTENSE_BOLD "\033[1;92m"

Definition at line 46 of file host.h.

Referenced by color_green_intense_bold().

5.33.1.15 #define MAGMA_COLOR_GREEN_UNDERLINE "\033[4;32m"

Definition at line 30 of file host.h.

Referenced by color_green_underline().

5.33.1.16 #define MAGMA_COLOR_PURPLE "\033[0;35m"

Definition at line 17 of file host.h.

Referenced by color_purple().

5.33.1.17 #define MAGMA_COLOR_PURPLE_BOLD "\033[1;35m"

Definition at line 25 of file host.h.

Referenced by color_purple_bold().

5.33.1.18 #define MAGMA_COLOR_PURPLE_INTENSE "\033[0;95m"

Definition at line 41 of file host.h.

Referenced by color_purple_intense().

5.33.1.19 #define MAGMA_COLOR_PURPLE_INTENSE_BOLD "\033[1;95m"

Definition at line 49 of file host.h.

Referenced by color_purple_intense_bold().

5.33.1.20 #define MAGMA_COLOR_PURPLE_UNDERLINE "\033[4;35m"

Definition at line 33 of file host.h.

Referenced by color_purple_underline().

5.33.1.21 #define MAGMA_COLOR_RED "\033[0;31m"

Definition at line 13 of file host.h.

Referenced by color_red().

5.33.1.22 #define MAGMA_COLOR_RED_BOLD "\033[1;31m"

Definition at line 21 of file host.h.

Referenced by color_red_bold().

5.33.1.23 #define MAGMA_COLOR_RED_INTENSE "\033[0;91m"

Definition at line 37 of file host.h.

Referenced by color_red_intense().

5.33.1.24 #define MAGMA_COLOR_RED_INTENSE_BOLD "\033[1;91m"

Definition at line 45 of file host.h.

Referenced by color_red_intense_bold().

5.33.1.25 #define MAGMA_COLOR_RED_UNDERLINE "\033[4;31m"

Definition at line 29 of file host.h.

Referenced by color_red_underline().

5.33.1.26 #define MAGMA_COLOR_RESET "\033[m"

Definition at line 11 of file host.h.

Referenced by color_reset().

5.33.1.27 #define MAGMA_COLOR_WHITE "\033[0;37m"

Definition at line 19 of file host.h.

Referenced by color_white().

5.33.1.28 #define MAGMA_COLOR_WHITE_BOLD "\033[1;37m"

Definition at line 27 of file host.h.

Referenced by color_white_bold().

5.33.1.29 #define MAGMA_COLOR_WHITE_INTENSE "\033[0;97m"

Definition at line 43 of file host.h.

Referenced by color_white_intense().

5.33.1.30 #define MAGMA_COLOR_WHITE_INTENSE_BOLD "\033[1;97m"

Definition at line 51 of file host.h.

Referenced by color_white_intense_bold().

5.33.1.31 #define MAGMA_COLOR_WHITE_UNDERLINE "\033[4;37m"

Definition at line 35 of file host.h.

Referenced by color_white_underline().

5.33.1.32 #define MAGMA_COLOR_YELLOW "\033[0;33m"

Definition at line 15 of file host.h.

Referenced by color_yellow().

5.33.1.33 #define MAGMA_COLOR_YELLOW_BOLD "\033[1;33m"

Definition at line 23 of file host.h.

Referenced by color_yellow_bold().

5.33.1.34 #define MAGMA_COLOR_YELLOW_INTENSE "\033[0;93m"

Definition at line 39 of file host.h.

Referenced by color_yellow_intense().

5.33.1.35 #define MAGMA_COLOR_YELLOW_INTENSE_BOLD "\033[1;93m"

Definition at line 47 of file host.h.

Referenced by color_yellow_intense_bold().

5.33.1.36 #define MAGMA_COLOR_YELLOW_UNDERLINE "\033[4;93m"

Definition at line 31 of file host.h.

Referenced by color_yellow_underline().

5.33.1.37 #define MAGMA_PROC_PATH "/proc"

Definition at line 53 of file host.h.

Referenced by process_find_pid(), and process_kill().

5.33.2 Typedef Documentation**5.33.2.1 octet_t**

Definition at line 58 of file host.h.

5.33.2.2 segment_t

Definition at line 63 of file host.h.

5.33.3 Enumeration Type Documentation**5.33.3.1 anonymous enum**

Enumerator:

MAGMA_SPOOL_BASE

MAGMA_SPOOL_DATA

MAGMA_SPOOL_SCAN

Definition at line 87 of file host.h.

5.33.4 Function Documentation

5.33.4.1 `int_t backtrace_print (void)`

[backtrace.c](#) [backtrace.c](#)

Note:

This function was created because `backtrace_symbols()` can fail due to heap corruption.

Returns:

-1 if the backtrace failed, or 0 on success.

Definition at line 15 of file backtrace.c.

5.33.4.2 `const chr_t* color_blue (void)`

[color.c](#)

Definition at line 44 of file color.c.

References `color_supported()`, and `MAGMA_COLOR_BLUE`.

5.33.4.3 `const chr_t* color_blue_bold (void)`

Definition at line 72 of file color.c.

References `color_supported()`, and `MAGMA_COLOR_BLUE_BOLD`.

5.33.4.4 `const chr_t* color_blue_intense (void)`

Definition at line 128 of file color.c.

References `color_supported()`, and `MAGMA_COLOR_BLUE_INTENSE`.

5.33.4.5 `const chr_t* color_blue_intense_bold (void)`

Definition at line 156 of file color.c.

References `color_supported()`, and `MAGMA_COLOR_BLUE_INTENSE_BOLD`.

5.33.4.6 `const chr_t* color_blue_underline (void)`

Definition at line 100 of file color.c.

References `color_supported()`, and `MAGMA_COLOR_BLUE_UNDERLINE`.

5.33.4.7 `const chr_t* color_cyan (void)`

Definition at line 52 of file color.c.

References `color_supported()`, and `MAGMA_COLOR_CYAN`.

5.33.4.8 const chr_t* color_cyan_bold (void)

Definition at line 80 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN_BOLD.

5.33.4.9 const chr_t* color_cyan_intense (void)

Definition at line 136 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN_INTENSE.

5.33.4.10 const chr_t* color_cyan_intense_bold (void)

Definition at line 164 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN_INTENSE_BOLD.

5.33.4.11 const chr_t* color_cyan_underline (void)

Definition at line 108 of file color.c.

References color_supported(), and MAGMA_COLOR_CYAN_UNDERLINE.

5.33.4.12 const chr_t* color_green (void)

Definition at line 36 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN.

5.33.4.13 const chr_t* color_green_bold (void)

Definition at line 64 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN_BOLD.

5.33.4.14 const chr_t* color_green_intense (void)

Definition at line 120 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN_INTENSE.

5.33.4.15 const chr_t* color_green_intense_bold (void)

Definition at line 148 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN_INTENSE_BOLD.

5.33.4.16 const chr_t* color_green_underline (void)

Definition at line 92 of file color.c.

References color_supported(), and MAGMA_COLOR_GREEN_UNDERLINE.

5.33.4.17 const chr_t* color_purple (void)

Definition at line 48 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE.

5.33.4.18 const chr_t* color_purple_bold (void)

Definition at line 76 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE_BOLD.

5.33.4.19 const chr_t* color_purple_intense (void)

Definition at line 132 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE_INTENSE.

5.33.4.20 const chr_t* color_purple_intense_bold (void)

Definition at line 160 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE_INTENSE_BOLD.

5.33.4.21 const chr_t* color_purple_underline (void)

Definition at line 104 of file color.c.

References color_supported(), and MAGMA_COLOR_PURPLE_UNDERLINE.

5.33.4.22 const chr_t* color_red (void)

Definition at line 32 of file color.c.

References color_supported(), and MAGMA_COLOR_RED.

5.33.4.23 const chr_t* color_red_bold (void)

Definition at line 60 of file color.c.

References color_supported(), and MAGMA_COLOR_RED_BOLD.

5.33.4.24 const chr_t* color_red_intense (void)

Definition at line 116 of file color.c.

References color_supported(), and MAGMA_COLOR_RED_INTENSE.

5.33.4.25 const chr_t* color_red_intense_bold (void)

Definition at line 144 of file color.c.

References color_supported(), and MAGMA_COLOR_RED_INTENSE_BOLD.

5.33.4.26 const chr_t* color_red_underline (void)

Definition at line 88 of file color.c.

References color_supported(), and MAGMA_COLOR_RED_UNDERLINE.

5.33.4.27 const chr_t* color_reset (void)

Definition at line 28 of file color.c.

References color_supported(), and MAGMA_COLOR_RESET.

5.33.4.28 bool_t color_supported (void)

Definition at line 10 of file color.c.

References magma_t::daemonize, magma, NULLER, st_cmp_ci_eq(), and magma_t::system.

Referenced by color_blue(), color_blue_bold(), color_blue_intense(), color_blue_intense_bold(), color_blue_underline(), color_cyan(), color_cyan_bold(), color_cyan_intense(), color_cyan_intense_bold(), color_cyan_underline(), color_green(), color_green_bold(), color_green_intense(), color_green_intense_bold(), color_green_underline(), color_purple(), color_purple_bold(), color_purple_intense(), color_purple_intense_bold(), color_purple_underline(), color_red(), color_red_bold(), color_red_intense(), color_red_intense_bold(), color_red_underline(), color_reset(), color_white(), color_white_bold(), color_white_intense(), color_white_intense_bold(), color_white_underline(), color_yellow(), color_yellow_bold(), color_yellow_intense(), color_yellow_intense_bold(), and color_yellow_underline().

5.33.4.29 const chr_t* color_white (void)

Definition at line 56 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE.

5.33.4.30 const chr_t* color_white_bold (void)

Definition at line 84 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE_BOLD.

5.33.4.31 const chr_t* color_white_intense (void)

Definition at line 140 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE_INTENSE.

5.33.4.32 const chr_t* color_white_intense_bold (void)

Definition at line 168 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE_INTENSE_BOLD.

5.33.4.33 const chr_t* color_white_underline (void)

Definition at line 112 of file color.c.

References color_supported(), and MAGMA_COLOR_WHITE_UNDERLINE.

5.33.4.34 const chr_t* color_yellow (void)

Definition at line 40 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW.

5.33.4.35 const chr_t* color_yellow_bold (void)

Definition at line 68 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW_BOLD.

5.33.4.36 const chr_t* color_yellow_intense (void)

Definition at line 124 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW_INTENSE.

5.33.4.37 const chr_t* color_yellow_intense_bold (void)

Definition at line 152 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW_INTENSE_BOLD.

5.33.4.38 const chr_t* color_yellow_underline (void)

Definition at line 96 of file color.c.

References color_supported(), and MAGMA_COLOR_YELLOW_UNDERLINE.

5.33.4.39 chr_t* errno_name (int *error*)

errors.c

Definition at line 181 of file errors.c.

References MAGMA_E2BIG, MAGMA_EACCES, MAGMA_EAGAIN, MAGMA_EBADF, MAGMA_EBUSY, MAGMA_ECHILD, MAGMA_EEXIST, MAGMA_EFAULT, MAGMA_EFBIG, MAGMA_EINTR, MAGMA_EINVAL, MAGMA_EIO, MAGMA_EISDIR, MAGMA_EMFILE, MAGMA_EMLINK, MAGMA_ENFILE, MAGMA_ENODEV, MAGMA_ENOENT, MAGMA_ENOEXEC, MAGMA_ENOMEM, MAGMA_ENOSPC, MAGMA_ENOTBLK, MAGMA_ENOTDIR, MAGMA_ENOTTY, MAGMA_ENXIO, MAGMA_EPERM, MAGMA_EPIPE, MAGMA_EROFS, MAGMA_ESPIPE, MAGMA_ESRCH, MAGMA_ETXTBSY, and MAGMA_EXDEV.

Referenced by tcp_continue(), tls_continue(), and tls_server_alloc().

5.33.4.40 bool_t file_accessible (const chr_t * *path*)

Determine whether a given filename or directory path is accessible by a user.

Parameters:

path a null-terminated string with the name of the filename or directory to be checked for existence/access.

Returns:

true if the pathname exists and is readable, or false otherwise.

Definition at line 155 of file files.c.

5.33.4.41 stringer_t* file_load (const char * *name*)

[files.c files.c](#)

Parameters:

name a character pointer to the full pathname of the file to be opened.

Returns:

NULL on failure, or a managed string containing all the data in the file.

Definition at line 51 of file files.c.

References [log_info](#), [st_alloc\(\)](#), [st_avail_get\(\)](#), [st_data_get\(\)](#), and [st_length_set\(\)](#).

Referenced by [config_load_file_settings\(\)](#), [dkim_start\(\)](#), and [prime_start\(\)](#).

5.33.4.42 int_t file_read (const char * *name*, stringer_t * *output*)

Get the contents of a file on disk.

See also:

[file_load\(\)](#)

Note:

This is similar in function to [file_load\(\)](#) but there is no [read\(\)](#) length checking, so it can be used on non-regular files.

Parameters:

name a character pointer to the full pathname of the file to be opened.

output a managed string where the results of the file read operation will be stored.

Returns:

-1 on failure or the number of bytes read from the file on success.

Definition at line 18 of file files.c.

References [log_info](#), [log_pedantic](#), [MEMORYBUF](#), [st_avail_get\(\)](#), [st_data_get\(\)](#), and [st_length_set\(\)](#).

Referenced by [process_find_pid\(\)](#), and [process_kill\(\)](#).

5.33.4.43 bool_t file_readwritable (const chr_t * *path*)

Determine whether a given filename or directory path is readable and writable by a user.

Parameters:

path a null-terminated string with the name of the filename or directory to be checked for permissions.

Returns:

true if the pathname exists and is readable and writable, or false otherwise.

Definition at line 165 of file files.c.

Referenced by [servers_validate\(\)](#).

5.33.4.44 int_t file_temp_handle (chr_t * *pdir*, stringer_t ** *tmpname*)

Get a file handle to a temporary file created in a specified directory.

Parameters:

pdir the parent directory in which to create the temporary file, or NULL for the default spool dir.
tmpname an optional pointer to a managed string to receive the name of the created temp file.

Returns:

-1 on failure or the new temporary file's file descriptor on success.

Definition at line 98 of file files.c.

References CONSTANT, HEAP, JOINTED, log_pedantic, MAGMA_SPOOL_BASE, MANAGED_T, ns_length_get(), spool_path(), st_append, st_char_get(), st_dupe(), st_dupe_opts(), st_free(), and st_import().

5.33.4.45 bool_t file_world_accessible (const chr_t * *path*)

Determine whether a given filename or directory path is readable or writable by other users.

Parameters:

path a null-terminated string with the name of the filename or directory to be checked for world permissions.

Returns:

true if the pathname exists and is readable or writable by others, or false otherwise.

Definition at line 175 of file files.c.

Referenced by dkim_start(), prime_start(), and servers_validate().

5.33.4.46 int_t folder_count (stringer_t * *path*, bool_t *recursive*, bool_t *strict*)

[folder.c](#)

Definition at line 39 of file folder.c.

References count, folder_count(), log_info, MEMORYBUF, NULLER, PLACER, st_char_get(), st_cmp_cs_ends(), st_cmp_cs_eq(), st_empty, st_free(), and st_merge.

Referenced by folder_count().

5.33.4.47 int_t folder_exists (stringer_t * *path*, bool_t *create*)

Check to see if a specified directory exists, or if specified, create it if it doesn't exist.

Parameters:

path a managed string containing the full pathname of the directory.
create if true, attempt to create the directory if it does not already exist.

Returns:

-1 if the directory doesn't exist or couldn't be created, 0 if the directory exists, or 1 if the directory was created.

Definition at line 16 of file folder.c.

References st_char_get().

Referenced by log_start(), and spool_check().

5.33.4.48 stringer_t* host_platform (stringer_t * *output*)

[host.c](#) [host.c](#)

Parameters:

output a pointer to a managed string to receive the OS description.

Returns:

NULL on failure, or the user-specified managed string containing the OS info on success.

Definition at line 15 of file host.c.

References `log_pedantic`, `ns_length_get()`, `st_avail_get()`, and `st_sprint()`.

Referenced by `lib_load()`.

5.33.4.49 stringer_t* host_version (stringer_t * *output*)

Get release information about the local OS.

Parameters:

output a pointer to a managed string to receive the release information.

Returns:

NULL on failure, or the user-specified managed string containing the release info on success.

Definition at line 41 of file host.c.

References `log_pedantic`, `ns_length_get()`, `st_avail_get()`, and `st_sprint()`.

Referenced by `lib_load()`.

5.33.4.50 bool_t ip_addr_eq (ip_t * *ip1*, ip_t * *ip2*)

[ip.c](#) [ip.c](#)

Parameters:

ip1 a pointer to the first ip address object in the comparison.

ip2 a pointer to the second ip address object in the comparison.

Returns:

true if the two addresses are equal, or false if they are not.

Definition at line 485 of file ip.c.

References `ip_t::family`, `ip_t::ip4`, and `ip_t::ip6`.

Referenced by `sess_get()`.

5.33.4.51 bool_t ip_addr_st (chr_t * *ipstr*, ip_t * *out*)

Convert a string into an IP address object.

Parameters:

ipstr a pointer to a null-terminated string containing the IP string to be parsed.

out a pointer to an IP address object that will receive result of the operation.

Returns:

true if the input string was a valid IP address or false if it was unable to be parsed.

Definition at line 507 of file ip.c.

References ip_t::family, ip_t::ip4, ip_t::ip6, and ns_length_get().

Referenced by ip_subnet_st().

5.33.4.52 ip_t* ip_copy (ip_t * dst, ip_t * src)

Create a copy of an IP address object.

Parameters:

dst a pointer to the destination IP address object to receive the copy.

src a pointer to the source IP address object to be copied.

NULL on failure, or a pointer to the destination IP address buffer on success.

Definition at line 153 of file ip.c.

References mm_copy().

Referenced by ip_subnet_st(), and sess_create().

5.33.4.53 int_t ip_family (ip_t * address)

An abstract method of retrieving the address family for an IP structure.

Returns:

returns a value between AF_UNSPEC (typically defined as 0) and AF_MAX, if an invalid family value is encountered outside this range, then it will be replaced with AF_UNSPEC. If an invalid address structure is provided, -1 will be returned.

Definition at line 17 of file ip.c.

References ip_t::family.

Referenced by ip_type().

5.33.4.54 bool_t ip_localhost (ip_t * address)

Determine whether an IP address matches the localhost loopback address.

Note:

This function only matches if the remote peer is connected using IPv4 from the 127.0.0.0/8 range, from the same range using an IPv4 to IPv6 mapping, or from the specific IPv6 using IPv6 from the ::1/128 address by IPv6.

Remarks:

The rationale behind this function is to make it easy for us to detect connections from the local host, and thus dictating that the packets associated with the connection never travel across a network cable, which allows us to exempt the peer from transport security requirements, SPF checks. and allow the use of administrative functions without authentication. In theory connections from IP addresses associated with the network interfaces on the local host are just as safe, but we currently only match against the loopback address.

Returns:

true if the connection appears to be using the loopback adapter, or false, if the connection appears to be from anywhere else.

Definition at line 75 of file ip.c.

References ip_t::ip6, ip_octet(), ip_type(), and ip_word().

Referenced by con_localhost(), con_private(), and ip_private().

5.33.4.55 bool_t ip_matches_subnet (subnet_t * *subnet*, ip_t * *addr*)

Determines whether a given IP address matches a subnet mask.

Parameters:

subnet a pointer to a subnet that the address will be compared against.

addr a pointer to the IP address of interest.

Returns:

true if the specified IP address falls into the subnet, or false if it does not.

Definition at line 617 of file ip.c.

References subnet_t::address, ip_t::family, ip_t::ip, and subnet_t::mask.

Referenced by smtp_bypass_check().

5.33.4.56 octet_t ip_octet (ip_t * *address*, int_t *position*)

Extract a specified 8 bit octet from an IPv4 or IPv6 address.

Parameters:

address the IP address object to be examined.

position the zero-indexed (starting at least-significant byte) byte number of the IP address to be evaluated.

Returns:

-1 on failure, or the 16 bit-widened octet extracted from the supplied IP address.

Definition at line 168 of file ip.c.

References ip_t::family, ip_t::ip4, and ip_t::ip6.

Referenced by con_addr_octet(), ip_localhost(), and ip_private().

5.33.4.57 stringer_t* ip_presentation (ip_t * *address*, stringer_t * *output*)

Convert an IP address structure into a readable string.

Parameters:

address a pointer to the IP address to be displayed.

output a managed string to store the output, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to the managed string containing the IP address as text on success.

Definition at line 296 of file ip.c.

References ip_t::family, ip_t::ip4, ip_t::ip6, log_pedantic, MEMORYBUF, ns_length_get(), st_alloc(), st_avail_get(), st_char_get(), st_free(), st_length_set(), st_valid_destination(), and st_valid_tracked().

Referenced by client_read(), client_read_line(), client_write(), con_addr_presentation(), net_trigger(), and tcp_addr_st().

5.33.4.58 bool_t ip_private (ip_t * address)

Determine whether an IP address appears to be associated with a non-routeable, private address space.

Note:

This function looks for the IP addresses in the address ranges set aside for private networks. For IPv4 the ranges 10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16 are reserved. For IPv6 the range fc00::/7 has been set aside. By convention addresses in these ranges are unroutable on the open internet, so if a peer appears to be connecting from a private address then it should be on the same private network.

Remarks:

The rationale behind this function is to make it easy for us to detect connections from hosts on the same (internal) network, and thus treat those connections differently than connections from the public internet.

Returns:

false by default, and true if the address falls inside the ranges reserved for private networks.

Definition at line 106 of file ip.c.

References ip_t::ip6, ip_localhost(), ip_octet(), ip_type(), and ip_word().

5.33.4.59 stringer_t* ip_reversed (ip_t * address, stringer_t * output)

Get a reversed-IP string, for use in an RBL lookup.

Parameters:

address a pointer to the IP address to be displayed as a string.

output a pointer to the managed string to receive the reversed-IP string, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to a managed string containing the reversed-IP string on success.

Definition at line 413 of file ip.c.

References ip_t::family, ip_t::ip4, ip_t::ip6, log_pedantic, st_alloc(), st_avail_get(), st_char_get(), st_length_set(), st_sprint(), st_valid_destination(), and st_valid_tracked().

Referenced by con_addr_reversed().

5.33.4.60 segment_t ip_segment (ip_t * address, int_t position)

Extract a specified 16 bit segment from an IPv4 or IPv6 address.

Parameters:

address the IP address object to be examined.

position the 0-indexed (starting at least-significant word) 16-bit segment of the IP address to be evaluated.

Returns:

-1 on failure, or the 32 bit-widened segment extracted from the supplied IP address.

Definition at line 193 of file ip.c.

References `ip_t::family`, `ip_t::ip4`, and `ip_t::ip6`.

Referenced by `con_addr_segment()`.

5.33.4.61 stringer_t* ip_standard (ip_t * address, stringer_t * output)

Convert an IP address structure to string representation.

Parameters:

address a pointer to the IP address to be converted.

output a pointer to the managed string to receive the converted value, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to a managed string containing a textual representation of the IP address.

Definition at line 339 of file ip.c.

References `ip_t::family`, `ip_t::ip4`, `ip_t::ip6`, `log_pedantic`, `st_alloc()`, `st_avail_get()`, `st_char_get()`, `st_free()`, `st_length_set()`, `st_sprint()`, `st_valid_destination()`, `st_valid_tracked()`, and `st_wipe()`.

Referenced by `con_addr_standard()`.

5.33.4.62 stringer_t* ip_subnet (ip_t * address, stringer_t * output)

Display the simple subnet string for an IP address.

Note:

IPv4 addresses will yield a /24 subnet address, and IPv6 addresses will result in a /64 subnet address. Neither address will be displayed with the trailing zero-octet(s).

Parameters:

address a pointer to the ip address to be examined.

output a pointer to a managed string to receive the subnet string, or if passed as NULL, it will be allocated for the caller.

Returns:

NULL on failure, or a pointer to a managed string containing the result on success.

Definition at line 243 of file ip.c.

References `ip_t::family`, `ip_t::ip4`, `ip_t::ip6`, `log_pedantic`, `st_alloc()`, `st_avail_get()`, `st_free()`, `st_length_get()`, `st_sprint()`, `st_valid_avail()`, and `st_valid_destination()`.

Referenced by `con_addr_subnet()`.

5.33.4.63 bool_t ip_subnet_st (chr_t * substr, subnet_t * out)

Convert a string to an IP address object.

Parameters:

ipstr a pointer to a null-terminated string containing the IP string to be parsed.

out a pointer to an IP address object that will receive result of the operation.

Returns:

true if the input string was a valid IP address or false if it was unable to be parsed.

Definition at line 557 of file ip.c.

References subnet_t::address, ip_t::family, ip_addr_st(), ip_copy(), subnet_t::mask, ns_free(), ns_import(), ns_length_get(), NULLER, pl_char_get(), pl_length_get(), placer_t, tok_get_count_st(), tok_get_ns(), and uint8_conv_bl().

Referenced by smtp_add_bypass_entry().

5.33.4.64 int8_t ip_type (ip_t * address)

Determine whether a structure holds an IPv4 or IPv6 address.

Returns:

returns -1 for invalid inputs, or system errors, 0 if the function isn't provided with a TCP/IP address, the number 4 if the address is IPv4 and 6 if the address is IPv6.

Definition at line 37 of file ip.c.

References ip_family().

Referenced by ip_localhost(), and ip_private().

5.33.4.65 uint32_t ip_word (ip_t * address, int_t position)

Get a specified 32-bit segment of an IP address.

Note:

This function operates on both ipv4 and ipv6 addresses.

Parameters:

address a pointer to the IP address object to be examined.

position a zero-based index into the 32-bit word(s) that comprise the target IP address.

Returns:

0 on error, or the specified 32-bit segment of the passed address.

Definition at line 221 of file ip.c.

References ip_t::family, ip_t::ip4, and ip_t::ip6.

Referenced by con_addr_word(), ip_localhost(), and ip_private().

5.33.4.66 pid_t process_find_pid (stringer_t * name)

process.c process.c

Note:

Only returns the first match, and will never return the process identifier for the currently running process.

Parameters:

name the name of the executable to find.

Returns:

0 if the process was not found, otherwise the a process identifier is returned.

Definition at line 23 of file process.c.

References chr_numeric(), file_read(), int32_conv_ns(), log_pedantic, MAGMA_FILEPATH_MAX, MAGMA_PROC_PATH, MANAGED-BUF, MEMORYBUF, pid, process_my_pid(), st_cmp_cs_eq(), and st_trim().

Referenced by main().

5.33.4.67 int_t process_kill (stringer_t * name, int_t signum, int_t wait)

Kill a named process with the specified signal, and retry if necessary.

Parameters:

name the name of the process to kill (matches any process that starts with the specified name).

signum the signal number to be sent to the matching process.

wait the number of times to re-send the signal, with one second rest intervals in between.

Returns:

-2 for a generic failure, -1 on timeout, 0 if process not found, and 1 if the process was successfully killed.

Definition at line 61 of file process.c.

References chr_numeric(), file_read(), int32_conv_ns(), log_pedantic, MAGMA_FILEPATH_MAX, MAGMA_PROC_PATH, MANAGED-BUF, MEMORYBUF, pid, process_my_pid(), st_char_get(), st_cmp_cs_starts(), st_length_int(), and st_swap().

5.33.4.68 pid_t process_my_pid (void)

Return the current process identifier using appropriate function for the current system.

Definition at line 13 of file process.c.

Referenced by process_find_pid(), process_kill(), and status_process().

5.33.4.69 chr_t* signal_name (int signal, char * buffer, size_t length)

[signals.c signals.c](#)

Parameters:

signal the input signal number.

buffer a buffer to receive the signal description.

length the length of the buffer to receive the output, which should be greater than 32 bytes.

Returns:

a pointer to a null-terminated string containing the textual name of the specified signal.

Definition at line 17 of file signals.c.

References log_pedantic, and SIGUNUSED.

Referenced by signal_segfault(), signal_shutdown(), and signal_status().

5.33.4.70 int_t spool_check (stringer_t * *path*)[spool.c](#) [spool.c](#)**Parameters:**

path a managed string with the spool directory to be checked.

Returns:

0 if the specified spool path exists, 1 if it was created, or -1 on error.

Definition at line 75 of file `spool.c`.

References `folder_exists()`, `log_critical`, `log_info`, `mutex_lock()`, `mutex_unlock()`, `st_char_get()`, and `st_length_int()`.

Referenced by `spool_mktemp()`, `spool_start()`, and `virus_start()`.

5.33.4.71 int_t spool_check_file (const char * *file*, const struct stat * *info*, int *type*)

An internal function used by the `ftw()` function to cleanup the file contents of a spool directory.

Parameters:

file a pointer to a null-terminated string containing the pathname of the spool file.

info a pointer to a stat object containing the filesystem info of the specified file.

the `ftw` type flag of the specified file (only `FTW_F` is handled).

Returns:

This function always returns 0.

Definition at line 185 of file `spool.c`.

References `log_error`, `MEMORYBUF`, `mutex_lock()`, `mutex_unlock()`, `NULLER`, `PLACER`, and `st_cmp_cs_eq()`.

Referenced by `spool_cleanup()`, and `spool_stop()`.

5.33.4.72 int_t spool_cleanup (void)

Clean up the file contents in the spool base directory.

Returns:

-1 on error or the number of files that were purged from the spool base directory.

Definition at line 213 of file `spool.c`.

References `log_error`, `log_pedantic`, `MAGMA_SPOOL_BASE`, `rwlock_lock_write()`, `rwlock_unlock()`, `spool_check_file()`, `spool_path()`, `st_char_get()`, `st_free()`, and `st_length_int()`.

Referenced by `spool_start()`.

5.33.4.73 uint64_t spool_error_stats (void)

Get the number of spool errors encountered.

Returns:

the total number of spool errors.

Definition at line 26 of file spool.c.

References mutex_lock(), and mutex_unlock().

Referenced by stats_derived_value().

5.33.4.74 int_t spool_mktemp (int_t *spool*, chr_t * *prefix*)

Create a temporary file in a specified spool directory.

Note:

The temp file is automatically unlinked as soon as it is created.

Parameters:

spool the spool directory id in which the temp file will be stored (MAGMA_SPOOL_BASE, MAGMA_SPOOL_DATA, or MAGMA_SPOOL_SCAN).

prefix an optional prefix for the temp file name ("magma" will be used if prefix is NULL0).

Returns:

-1 on failure or the file descriptor to the newly created temp file on success.

Definition at line 113 of file spool.c.

References log_critical, log_pedantic, MAGMA_SPOOL_BASE, MEMORYBUF, mutex_lock(), mutex_unlock(), rand_get_uint64(), rwlock_lock_read(), rwlock_unlock(), spool_check(), spool_path(), st_aprint(), st_char_get(), st_free(), st_length_int(), and thread_get_thread_id().

Referenced by st_alloc_opts(), and virus_check().

5.33.4.75 stringer_t* spool_path (int_t *spool*)

Get the full path to a requested spool directory.

Parameters:

the spool id (MAGMA_SPOOL_BASE, MAGMA_SPOOL_DATA, or MAGMA_SPOOL_SCAN); defaults to MAGMA_SPOOL_DATA.

Returns:

NULL on failure, or a managed string pointing to the requested path inside the magma spool parent directory, or /tmp/magma if the spool isn't configured.

Definition at line 40 of file spool.c.

References CONTIGUOUS, HEAP, magma, MAGMA_SPOOL_BASE, MAGMA_SPOOL_SCAN, ns_length_get(), NULLER_T, magma_t::spool, and st_merge_opts().

Referenced by dspam_check(), dspam_train(), file_temp_handle(), spool_cleanup(), spool_mktemp(), spool_start(), and virus_start().

5.33.4.76 bool_t spool_start (void)

Create and check the spool directories, and clean them for use.

Returns:

true on success or false on failure.

Definition at line 279 of file `spool.c`.

References `log_critical`, `MAGMA_SPOOL_BASE`, `MAGMA_SPOOL_DATA`, `MAGMA_SPOOL_SCAN`, `mutex_lock()`, `mutex_unlock()`, `spool_check()`, `spool_cleanup()`, `spool_path()`, `st_char_get()`, `st_free()`, and `st_length_int()`.

Referenced by `process_start()`.

5.33.4.77 `void spool_stop (void)`

Clean up the file contents in the spool base directory.

Note:

This is like a fast version of `spool_cleanup()`.

Returns:

This function returns no value.

Definition at line 252 of file `spool.c`.

References `log_pedantic`, `spool_check_file()`, `st_char_get()`, `st_free()`, and `st_length_int()`.

Referenced by `process_stop()`.

5.33.4.78 `ip_t* tcp_addr_ip (int sockd, ip_t * output)`

[tcp.c](#)

Definition at line 163 of file `tcp.c`.

References `ip_t::family`, `ip_t::ip4`, `ip_t::ip6`, `MEMORYBUF`, `mm_alloc()`, and `mm_copy()`.

Referenced by `con_init()`, and `tcp_addr_st()`.

5.33.4.79 `stringer_t* tcp_addr_st (int sockd, stringer_t * output)`

Definition at line 192 of file `tcp.c`.

References `ip_presentation()`, and `tcp_addr_ip()`.

5.33.4.80 `int tcp_continue (int sockd, int result, int syserror)`

Return -1 if the connection is invalid, 0 if the operation should be retried, or a positive number indicating the number of bytes processed.

Definition at line 84 of file `tcp.c`.

References `errno_name()`, `log_pedantic`, `MEMORYBUF`, and `status`.

Referenced by `tcp_read()`, and `tcp_write()`.

5.33.4.81 `int_t tcp_error (int error)`

Determine whether the error is a permanent/fatal failure, or a transient error.

Returns:

return 0 for errors we should ignore, and -1 for permanent/fatal failures.

Definition at line 15 of file `tcp.c`.

References log_pedantic, and MEMORYBUF.

Referenced by tcp_status().

5.33.4.82 int tcp_read (int sockd, void * buffer, int length, bool_t block)

Read data from a TCP/IP network socket.

Parameters:

sockd the socket file descriptor we'll read the data from.

buffer a pointer to the buffer where the data will be stored.

length the maximum length, in bytes, we can read into the buffer.

block a boolean to indicating whether to make a blocking write call.

Returns:

-1 if the network connection is no longer viable, otherwise the number bytes read, while a 0 may indicate the connection wasn't ready (when using non-blocking read calls), or a non-network error may have occurred.

Definition at line 109 of file tcp.c.

References log_pedantic, and tcp_continue().

Referenced by con_read(), and con_read_line().

5.33.4.83 int_t tcp_status (int sockd)

Determine whether the socket connection provided by sockd is still valid.

Parameters:

sockd The socket connection being checked.

Returns:

return 0 for valid connections

Definition at line 42 of file tcp.c.

References MANAGEDBUF, st_data_get(), and tcp_error().

Referenced by client_read(), client_read_line(), client_status(), client_write(), and con_status().

5.33.4.84 int tcp_wait (int sockd)

Blocks until a socket is ready for a read/write operation, or otherwise becomes invalid.

Parameters:

sockd

Returns:

Definition at line 75 of file tcp.c.

5.33.4.85 int tcp_write (int *sockd*, const void * *buffer*, int *length*, bool_t *block*)

Write data to an open TCP/IP network socket.

Parameters:

sockd the socket file descriptor we'll write the data to.

buffer a pointer to the buffer containing the data to be written.

length the length, in bytes, of the data to be written.

block a boolean to indicating whether to make a blocking write call.

Returns:

-1 on error, or the number of bytes written to the network connection.

Definition at line 140 of file tcp.c.

References `log_pedantic`, and `tcp_continue()`.

Referenced by `con_write_bl()`.

5.34 src/core/host/ip.c File Reference

Generic interface for parsing, and manipulating the IP address structure. `#include "magma.h"`

Functions

- `int_t ip_family (ip_t *address)`
An abstract method of retrieving the address family for an IP structure.
- `int8_t ip_type (ip_t *address)`
Determine whether a structure holds an IPv4 or IPv6 address.
- `bool_t ip_localhost (ip_t *address)`
Determine whether an IP address matches the localhost loopback address.
- `bool_t ip_private (ip_t *address)`
Determine whether an IP address appears to be associated with a non-routeable, private address space.
- `ip_t * ip_copy (ip_t *dst, ip_t *src)`
Create a copy of an IP address object.
- `octet_t ip_octet (ip_t *address, int_t position)`
Extract a specified 8 bit octet from an IPv4 or IPv6 address.
- `segment_t ip_segment (ip_t *address, int_t position)`
Extract a specified 16 bit segment from an IPv4 or IPv6 address.
- `uint32_t ip_word (ip_t *address, int_t position)`
Get a specified 32-bit segment of an IP address.
- `stringer_t * ip_subnet (ip_t *address, stringer_t *output)`
Display the simple subnet string for an IP address.
- `stringer_t * ip_presentation (ip_t *address, stringer_t *output)`
Convert an IP address structure into a readable string.
- `stringer_t * ip_standard (ip_t *address, stringer_t *output)`
Convert an IP address structure to string representation.
- `stringer_t * ip_reversed (ip_t *address, stringer_t *output)`
Get a reversed-IP string, for use in an RBL lookup.
- `bool_t ip_addr_eq (ip_t *ip1, ip_t *ip2)`
Determine whether two IP addresses are equal.
- `bool_t ip_addr_st (chr_t *ipstr, ip_t *out)`
Convert a string into an IP address object.
- `bool_t ip_subnet_st (chr_t *substr, subnet_t *out)`
Convert a string to an IP address object.
- `bool_t ip_matches_subnet (subnet_t *subnet, ip_t *addr)`
Determines whether a given IP address matches a subnet mask.

5.34.1 Detailed Description

Generic interface for parsing, and manipulating the IP address structure.

Definition in file [ip.c](#).

5.34.2 Function Documentation

5.34.2.1 `bool_t ip_addr_eq (ip_t * ip1, ip_t * ip2)`

Determine whether two IP addresses are equal. [ip.c](#)

Parameters:

- ip1* a pointer to the first ip address object in the comparison.
- ip2* a pointer to the second ip address object in the comparison.

Returns:

true if the two addresses are equal, or false if they are not.

Definition at line 485 of file [ip.c](#).

References [ip_t::family](#), [ip_t::ip4](#), and [ip_t::ip6](#).

Referenced by [sess_get\(\)](#).

5.34.2.2 `bool_t ip_addr_st (chr_t * ipstr, ip_t * out)`

Convert a string into an IP address object.

Parameters:

- ipstr* a pointer to a null-terminated string containing the IP string to be parsed.
- out* a pointer to an IP address object that will receive result of the operation.

Returns:

true if the input string was a valid IP address or false if it was unable to be parsed.

Definition at line 507 of file [ip.c](#).

References [ip_t::family](#), [ip_t::ip4](#), [ip_t::ip6](#), and [ns_length_get\(\)](#).

Referenced by [ip_subnet_st\(\)](#).

5.34.2.3 `ip_t* ip_copy (ip_t * dst, ip_t * src)`

Create a copy of an IP address object.

Parameters:

- dst* a pointer to the destination IP address object to receive the copy.
- src* a pointer to the source IP address object to be copied.
- NULL* on failure, or a pointer to the destination IP address buffer on success.

Definition at line 153 of file [ip.c](#).

References [mm_copy\(\)](#).

Referenced by [ip_subnet_st\(\)](#), and [sess_create\(\)](#).

5.34.2.4 `int_t ip_family (ip_t * address)`

An abstract method of retrieving the address family for an IP structure.

Returns:

returns a value between AF_UNSPEC (typically defined as 0) and AF_MAX, if an invalid family value is encountered outside this range, then it will be replaced with AF_UNSPEC. If an invalid address structure is provided, -1 will be returned.

Definition at line 17 of file ip.c.

References `ip_t::family`.

Referenced by `ip_type()`.

5.34.2.5 `bool_t ip_localhost (ip_t * address)`

Determine whether an IP address matches the localhost loopback address.

Note:

This function only matches if the remote peer is connected using IPv4 from the 127.0.0.0/8 range, from the same range using an IPv4 to IPv6 mapping, or from the specific IPv6 using IPv6 from the ::1/128 address by IPv6.

Remarks:

The rationale behind this function is to make it easy for us to detect connections from the local host, and thus dictating that the packets associated with the connection never travel across a network cable, which allows us to exempt the peer from transport security requirements, SPF checks. and allow the use of administrative functions without authentication. In theory connections from IP addresses associated with the network interfaces on the local host are just as safe, but we currently only match against the loopback address.

Returns:

true if the connection appears to be using the loopback adapter, or false, if the connection appears to be from anywhere else.

Definition at line 75 of file ip.c.

References `ip_t::ip6`, `ip_octet()`, `ip_type()`, and `ip_word()`.

Referenced by `con_localhost()`, `con_private()`, and `ip_private()`.

5.34.2.6 `bool_t ip_matches_subnet (subnet_t * subnet, ip_t * addr)`

Determines whether a given IP address matches a subnet mask.

Parameters:

subnet a pointer to a subnet that the address will be compared against.

addr a pointer to the IP address of interest.

Returns:

true if the specified IP address falls into the subnet, or false if it does not.

Definition at line 617 of file ip.c.

References `subnet_t::address`, `ip_t::family`, `ip_t::ip`, and `subnet_t::mask`.

Referenced by `sntp_bypass_check()`.

5.34.2.7 `octet_t ip_octet (ip_t * address, int_t position)`

Extract a specified 8 bit octet from an IPv4 or IPv6 address.

Parameters:

address the IP address object to be examined.

position the zero-indexed (starting at least-significant byte) byte number of the IP address to be evaluated.

Returns:

-1 on failure, or the 16 bit-widened octet extracted from the supplied IP address.

Definition at line 168 of file ip.c.

References ip_t::family, ip_t::ip4, and ip_t::ip6.

Referenced by con_addr_octet(), ip_localhost(), and ip_private().

5.34.2.8 `stringer_t* ip_presentation (ip_t * address, stringer_t * output)`

Convert an IP address structure into a readable string.

Parameters:

address a pointer to the IP address to be displayed.

output a managed string to store the output, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to the managed string containing the IP address as text on success.

Definition at line 296 of file ip.c.

References ip_t::family, ip_t::ip4, ip_t::ip6, log_pedantic, MEMORYBUF, ns_length_get(), st_alloc(), st_avail_get(), st_char_get(), st_free(), st_length_set(), st_valid_destination(), and st_valid_tracked().

Referenced by client_read(), client_read_line(), client_write(), con_addr_presentation(), net_trigger(), and tcp_addr_st().

5.34.2.9 `bool_t ip_private (ip_t * address)`

Determine whether an IP address appears to be associated with a non-routeable, private address space.

Note:

This function looks for the IP addresses in the address ranges set aside for private networks. For IPv4 the ranges 10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16 are reserved. For IPv6 the range fc00::/7 has been set aside. By convention addresses in these ranges are unrouteable on the open internet, so if a peer appears to be connecting from a private address then it should be on the same private network.

Remarks:

The rationale behind this function is to make it easy for us to detect connections from hosts on the same (internal) network, and thus treat those connections differently than connections from the public internet.

Returns:

false by default, and true if the address falls inside the ranges reserved for private networks.

Definition at line 106 of file ip.c.

References ip_t::ip6, ip_localhost(), ip_octet(), ip_type(), and ip_word().

5.34.2.10 stringer_t* ip_reversed(ip_t * address, stringer_t * output)

Get a reversed-IP string, for use in an RBL lookup.

Parameters:

address a pointer to the IP address to be displayed as a string.

output a pointer to the managed string to receive the reversed-IP string, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to a managed string containing the reversed-IP string on success.

Definition at line 413 of file ip.c.

References ip_t::family, ip_t::ip4, ip_t::ip6, log_pedantic, st_alloc(), st_avail_get(), st_char_get(), st_length_set(), st_sprint(), st_valid_destination(), and st_valid_tracked().

Referenced by con_addr_reversed().

5.34.2.11 segment_t ip_segment(ip_t * address, int_t position)

Extract a specified 16 bit segment from an IPv4 or IPv6 address.

Parameters:

address the IP address object to be examined.

position the 0-indexed (starting at least-significant word) 16-bit segment of the IP address to be evaluated.

Returns:

-1 on failure, or the 32 bit-widened segment extracted from the supplied IP address.

Definition at line 193 of file ip.c.

References ip_t::family, ip_t::ip4, and ip_t::ip6.

Referenced by con_addr_segment().

5.34.2.12 stringer_t* ip_standard(ip_t * address, stringer_t * output)

Convert an IP address structure to string representation.

Parameters:

address a pointer to the IP address to be converted.

output a pointer to the managed string to receive the converted value, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to a managed string containing a textual representation of the IP address.

Definition at line 339 of file ip.c.

References ip_t::family, ip_t::ip4, ip_t::ip6, log_pedantic, st_alloc(), st_avail_get(), st_char_get(), st_free(), st_length_set(), st_sprint(), st_valid_destination(), st_valid_tracked(), and st_wipe().

Referenced by con_addr_standard().

5.34.2.13 stringer_t* ip_subnet (ip_t * *address*, stringer_t * *output*)

Display the simple subnet string for an IP address.

Note:

IPv4 addresses will yield a /24 subnet address, and IPv6 addresses will result in a /64 subnet address. Neither address will be displayed with the trailing zero-octet(s).

Parameters:

address a pointer to the ip address to be examined.

output a pointer to a managed string to receive the subnet string, or if passed as NULL, it will be allocated for the caller.

Returns:

NULL on failure, or a pointer to a managed string containing the result on success.

Definition at line 243 of file ip.c.

References ip_t::family, ip_t::ip4, ip_t::ip6, log_pedantic, st_alloc(), st_avail_get(), st_free(), st_length_get(), st_sprint(), st_valid_avail(), and st_valid_destination().

Referenced by con_addr_subnet().

5.34.2.14 bool_t ip_subnet_st (chr_t * *substr*, subnet_t * *out*)

Convert a string to an IP address object.

Parameters:

ipstr a pointer to a null-terminated string containing the IP string to be parsed.

out a pointer to an IP address object that will receive result of the operation.

Returns:

true if the input string was a valid IP address or false if it was unable to be parsed.

Definition at line 557 of file ip.c.

References subnet_t::address, ip_t::family, ip_addr_st(), ip_copy(), subnet_t::mask, ns_free(), ns_import(), ns_length_get(), NULLER, pl_char_get(), pl_length_get(), placer_t, tok_get_count_st(), tok_get_ns(), and uint8_conv_bl().

Referenced by smtp_add_bypass_entry().

5.34.2.15 int8_t ip_type (ip_t * *address*)

Determine whether a structure holds an IPv4 or IPv6 address.

Returns:

returns -1 for invalid inputs, or system errors, 0 if the function isn't provided with a TCP/IP address, the number 4 if the address is IPv4 and 6 if the address is IPv6.

Definition at line 37 of file ip.c.

References ip_family().

Referenced by ip_localhost(), and ip_private().

5.34.2.16 uint32_t ip_word (ip_t * *address*, int_t *position*)

Get a specified 32-bit segment of an IP address.

Note:

This function operates on both ipv4 and ipv6 addresses.

Parameters:

address a pointer to the IP address object to be examined.

position a zero-based index into the 32-bit word(s) that comprise the target IP address.

Returns:

0 on error, or the specified 32-bit segment of the passed address.

Definition at line 221 of file ip.c.

References ip_t::family, ip_t::ip4, and ip_t::ip6.

Referenced by con_addr_word(), ip_localhost(), and ip_private().

5.35 src/core/host/process.c File Reference

```
#include "magma.h"
```

Functions

- `pid_t process_my_pid` (void)
Return the current process identifier using appropriate function for the current system.
- `pid_t process_find_pid` (stringer_t *name)
Find the process identifier for a named executable and return it.
- `int_t process_kill` (stringer_t *name, int_t signum, int_t wait)
Kill a named process with the specified signal, and retry if necessary.

5.35.1 Function Documentation

5.35.1.1 pid_t process_find_pid (stringer_t * name)

Find the process identifier for a named executable and return it. process.c

Note:

Only returns the first match, and will never return the process identifier for the currently running process.

Parameters:

name the name of the executable to find.

Returns:

0 if the process was not found, otherwise the a process identifier is returned.

Definition at line 23 of file process.c.

References `chr_numeric()`, `file_read()`, `int32_conv_ns()`, `log_pedantic`, `MAGMA_FILEPATH_MAX`, `MAGMA_PROC_PATH`, `MANAGED-BUF`, `MEMORYBUF`, `pid`, `process_my_pid()`, `st_cmp_cs_eq()`, and `st_trim()`.

Referenced by `main()`.

5.35.1.2 int_t process_kill (stringer_t * name, int_t signum, int_t wait)

Kill a named process with the specified signal, and retry if necessary.

Parameters:

name the name of the process to kill (matches any process that starts with the specified name).

signum the signal number to be sent to the matching process.

wait the number of times to re-send the signal, with one second rest intervals in between.

Returns:

-2 for a generic failure, -1 on timeout, 0 if process not found, and 1 if the process was successfully killed.

Definition at line 61 of file process.c.

References `chr_numeric()`, `file_read()`, `int32_conv_ns()`, `log_pedantic`, `MAGMA_FILEPATH_MAX`, `MAGMA_PROC_PATH`, `MANAGED-BUF`, `MEMORYBUF`, `pid`, `process_my_pid()`, `st_char_get()`, `st_cmp_cs_starts()`, `st_length_int()`, and `st_swap()`.

5.35.1.3 pid_t process_my_pid (void)

Return the current process identifier using appropriate function for the current system.

Definition at line 13 of file process.c.

Referenced by process_find_pid(), process_kill(), and status_process().

5.36 src/engine/context/process.c File Reference

```
#include "magma.h"
```

Functions

- void `process_maint` (void)
The entry point for the process maintenance thread, which runs in a continuous loop unless canceled.
- void `process_stop` (void)
Execute the magma shutdown routines.
- `bool_t` `process_start` (void)
Execute the magma initializations routines, making sure they all run properly.

Variables

- `bool_t` `exit_and_dump`
- `uint64_t` `day` = 0
- `pthread_t` * `maint` = NULL

5.36.1 Function Documentation

5.36.1.1 void process_maint (void)

The entry point for the process maintenance thread, which runs in a continuous loop unless canceled.

Note:

Execute once daily: rotate the log files, update the warehouse, and perform tank maintenance. Execute every few (0-10) minutes: refresh the virus engine and prune the object cache.

Returns:

This function returns no value.

Definition at line 20 of file process.c.

References `day`, `log_rotate()`, `obj_cache_prune()`, `rand_get_uint32()`, `status`, `thread_cancel_disable()`, `thread_cancel_enable()`, `thread_start()`, `thread_stop()`, `time_datestamp()`, `time_till_midnight()`, `virus_engine_refresh()`, and `warehouse_update()`.

Referenced by `process_start()`.

5.36.1.2 bool_t process_start (void)

Execute the magma initializations routines, making sure they all run properly.

Note:

If any of the init routines returns with failure, this function fails and logs an error message. Certain checks are made that the init routines are run in a particular order, especially in waiting for daemonization and privilege dropping before starting logging. The system maintenance thread is also launched from here.

Returns:

true if all starter functions returned true, or false if any of them failed..

Definition at line 163 of file process.c.

References `cache_start()`, `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_defaults()`, `config_load_file_settings()`, `config_validate_settings()`, `deprecated_ecies_start()`, `dkim_start()`, `dspam_start()`, `exit_and_dump`, `magma_t::host`, `http_content_start()`, `magma_t::init`, `lib_load()`, `log_critical`, `log_options`, `log_start()`, `M_LOG_FILE_DISABLE`, `M_LOG_FUNCTION_DISABLE`, `M_LOG_INFO`, `M_LOG_LINE_DISABLE`, `M_LOG_STACK_TRACE_DISABLE`, `M_LOG_TIME_DISABLE`, `magma`, `MAGMA_HOSTNAME_MAX`, `mail_cache_start()`, `maint`, `mm_alloc()`, `mm_free()`, `mm_sec_start()`, `magma_t::name`, `obj_cache_start()`, `magma_t::page_length`, `prime_start()`, `process_maint()`, `protocol_init()`, `queue_init()`, `rand_start()`, `sanity_check()`, `servers_encryption_start()`, `servers_network_start()`, `signal_start()`, `spf_start()`, `spool_start()`, `sql_start()`, `ssl_start()`, `stats_init()`, `status_process()`, `system_change_root_directory()`, `system_fork_daemon()`, `system_init_core_dumps()`, `system_init_impersonation()`, `system_init_resource_limits()`, `system_init_umask()`, `tank_start()`, `thread_launch()`, `virus_start()`, `warehouse_start()`, and `xml_start()`.

Referenced by `main()`.

5.36.1.3 void process_stop (void)

Execute the magma shutdown routines.

Note:

This function will execute a shutdown routine for each of the startup initialization routines that was performed. It will also signal and terminate the maintenance thread.

Returns:

This function returns no value.

Definition at line 66 of file process.c.

References `cache_stop()`, `config_free()`, `deprecated_ecies_stop()`, `dkim_stop()`, `dspam_stop()`, `http_content_stop()`, `magma_t::init`, `lib_unload()`, `log_critical`, `log_info`, `log_options`, `log_pedantic`, `M_LOG_FILE_DISABLE`, `M_LOG_FUNCTION_DISABLE`, `M_LOG_INFO`, `M_LOG_LINE_DISABLE`, `M_LOG_STACK_TRACE_DISABLE`, `M_LOG_TIME_DISABLE`, `magma`, `mail_cache_stop()`, `maint`, `mm_free()`, `mm_sec_stop()`, `obj_cache_stop()`, `prime_stop()`, `queue_shutdown()`, `rand_stop()`, `servers_encryption_stop()`, `servers_network_stop()`, `spf_stop()`, `spool_stop()`, `sql_stop()`, `ssl_stop()`, `stats_shutdown()`, `status`, `tank_stop()`, `thread_join()`, `thread_signal()`, `virus_stop()`, `warehouse_stop()`, and `xml_stop()`.

Referenced by `main()`.

5.36.2 Variable Documentation

5.36.2.1 uint64_t day = 0

Definition at line 11 of file process.c.

Referenced by `process_maint()`.

5.36.2.2 bool_t exit_and_dump

Definition at line 13 of file global.c.

Referenced by `args_parse()`, `config_validate_settings()`, and `process_start()`.

5.36.2.3 pthread_t* maint = NULL

Definition at line 12 of file process.c.

Referenced by `process_start()`, and `process_stop()`.

5.37 src/core/host/signals.c File Reference

Functions to help handle signals. `#include "magma.h"`

Functions

- `chr_t * signal_name` (int *signal*, char **buffer*, size_t *length*)

Translate a numeric signal into a human readable name.

5.37.1 Detailed Description

Functions to help handle signals.

Definition in file [signals.c](#).

5.37.2 Function Documentation

5.37.2.1 `chr_t* signal_name` (int *signal*, char * *buffer*, size_t *length*)

Translate a numeric signal into a human readable name. [signals.c](#)

Parameters:

signal the input signal number.

buffer a buffer to receive the signal description.

length the length of the buffer to receive the output, which should be greater than 32 bytes.

Returns:

a pointer to a null-terminated string containing the textual name of the specified signal.

Definition at line 17 of file [signals.c](#).

References [log_pedantic](#), and [SIGUNUSED](#).

Referenced by [signal_segfault\(\)](#), [signal_shutdown\(\)](#), and [signal_status\(\)](#).

5.38 src/core/host/spool.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t spool_error_stats` (void)
Get the number of spool errors encountered.
- `stringer_t * spool_path` (int_t spool)
Get the full path to a requested spool directory.
- `int_t spool_check` (stringer_t *path)
Check to see if the specified spool directory exists.
- `int_t spool_mktemp` (int_t spool, chr_t *prefix)
Create a temporary file in a specified spool directory.
- `int_t spool_check_file` (const char *file, const struct stat *info, int type)
An internal function used by the ftw() function to cleanup the file contents of a spool directory.
- `int_t spool_cleanup` (void)
Clean up the file contents in the spool base directory.
- `void spool_stop` (void)
Clean up the file contents in the spool base directory.
- `bool_t spool_start` (void)
Create and check the spool directories, and clean them for use.

5.38.1 Function Documentation

5.38.1.1 int_t spool_check (stringer_t * path)

Check to see if the specified spool directory exists. [spool.c](#)

Parameters:

path a managed string with the spool directory to be checked.

Returns:

0 if the specified spool path exists, 1 if it was created, or -1 on error.

Definition at line 75 of file spool.c.

References [folder_exists\(\)](#), [log_critical](#), [log_info](#), [mutex_lock\(\)](#), [mutex_unlock\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

Referenced by [spool_mktemp\(\)](#), [spool_start\(\)](#), and [virus_start\(\)](#).

5.38.1.2 `int_t spool_check_file (const char *file, const struct stat *info, int type)`

An internal function used by the `ftw()` function to cleanup the file contents of a spool directory.

Parameters:

- file* a pointer to a null-terminated string containing the pathname of the spool file.
- info* a pointer to a stat object containing the filesystem info of the specified file.
- the* ftw type flag of the specified file (only FTW_F is handled).

Returns:

This function always returns 0.

Definition at line 185 of file `spool.c`.

References `log_error`, `MEMORYBUF`, `mutex_lock()`, `mutex_unlock()`, `NULLER`, `PLACER`, and `st_cmp_cs_eq()`.

Referenced by `spool_cleanup()`, and `spool_stop()`.

5.38.1.3 `int_t spool_cleanup (void)`

Clean up the file contents in the spool base directory.

Returns:

-1 on error or the number of files that were purged from the spool base directory.

Definition at line 213 of file `spool.c`.

References `log_error`, `log_pedantic`, `MAGMA_SPOOL_BASE`, `rwlock_lock_write()`, `rwlock_unlock()`, `spool_check_file()`, `spool_path()`, `st_char_get()`, `st_free()`, and `st_length_int()`.

Referenced by `spool_start()`.

5.38.1.4 `uint64_t spool_error_stats (void)`

Get the number of spool errors encountered.

Returns:

the total number of spool errors.

Definition at line 26 of file `spool.c`.

References `mutex_lock()`, and `mutex_unlock()`.

Referenced by `stats_derived_value()`.

5.38.1.5 `int_t spool_mktemp (int_t spool, chr_t *prefix)`

Create a temporary file in a specified spool directory.

Note:

The temp file is automatically unlinked as soon as it is created.

Parameters:

- spool* the spool directory id in which the temp file will be stored (`MAGMA_SPOOL_BASE`, `MAGMA_SPOOL_DATA`, or `MAGMA_SPOOL_SCAN`).

prefix an optional prefix for the temp file name ("magma" will be used if prefix is NULL0).

Returns:

-1 on failure or the file descriptor to the newly created temp file on success.

Definition at line 113 of file spool.c.

References `log_critical`, `log_pedantic`, `MAGMA_SPOOL_BASE`, `MEMORYBUF`, `mutex_lock()`, `mutex_unlock()`, `rand_get_uint64()`, `rwlock_lock_read()`, `rwlock_unlock()`, `spool_check()`, `spool_path()`, `st_aprint()`, `st_char_get()`, `st_free()`, `st_length_int()`, and `thread_get_thread_id()`.

Referenced by `st_alloc_opts()`, and `virus_check()`.

5.38.1.6 `stringer_t* spool_path (int_t spool)`

Get the full path to a requested spool directory.

Parameters:

the spool id (`MAGMA_SPOOL_BASE`, `MAGMA_SPOOL_DATA`, or `MAGMA_SPOOL_SCAN`); defaults to `MAGMA_SPOOL_DATA`.

Returns:

NULL on failure, or a managed string pointing to the requested path inside the magma spool parent directory, or `/tmp/magma` if the spool isn't configured.

Definition at line 40 of file spool.c.

References `CONTIGUOUS`, `HEAP`, `magma`, `MAGMA_SPOOL_BASE`, `MAGMA_SPOOL_SCAN`, `ns_length_get()`, `NULLER_T`, `magma_t::spool`, and `st_merge_opts()`.

Referenced by `dspam_check()`, `dspam_train()`, `file_temp_handle()`, `spool_cleanup()`, `spool_mktemp()`, `spool_start()`, and `virus_start()`.

5.38.1.7 `bool_t spool_start (void)`

Create and check the spool directories, and clean them for use.

Returns:

true on success or false on failure.

Definition at line 279 of file spool.c.

References `log_critical`, `MAGMA_SPOOL_BASE`, `MAGMA_SPOOL_DATA`, `MAGMA_SPOOL_SCAN`, `mutex_lock()`, `mutex_unlock()`, `spool_check()`, `spool_cleanup()`, `spool_path()`, `st_char_get()`, `st_free()`, and `st_length_int()`.

Referenced by `process_start()`.

5.38.1.8 `void spool_stop (void)`

Clean up the file contents in the spool base directory.

Note:

This is like a fast version of `spool_cleanup()`.

Returns:

This function returns no value.

Definition at line 252 of file spool.c.

References `log_pedantic`, `spool_check_file()`, `st_char_get()`, `st_free()`, and `st_length_int()`.

Referenced by `process_stop()`.

5.39 src/core/host/tcp.c File Reference

Generic fuctions for interaction with TCP/IP socket connections. `#include "magma.h"`

Functions

- [int_t tcp_error](#) (int error)
Determine whether the error is a permanent/fatal failure, or a transient error.
- [int_t tcp_status](#) (int sockd)
Determine whether the socket connection provided by sockd is still valid.
- [int tcp_wait](#) (int sockd)
Blocks until a socket is ready for a read/write operation, or otherwise becomes invalid.
- [int tcp_continue](#) (int sockd, int result, int syserror)
Return -1 if the connection is invalid, 0 if the operation should be retried, or a positive number indicating the number of bytes processed.
- [int tcp_read](#) (int sockd, void *buffer, int length, bool_t block)
Read data from a TCP/IP network socket.
- [int tcp_write](#) (int sockd, const void *buffer, int length, bool_t block)
Write data to an open TCP/IP network socket.
- [ip_t * tcp_addr_ip](#) (int sockd, [ip_t](#) *output)
tcp.c
- [stringer_t * tcp_addr_st](#) (int sockd, [stringer_t](#) *output)

5.39.1 Detailed Description

Generic fuctions for interaction with TCP/IP socket connections.

Definition in file [tcp.c](#).

5.39.2 Function Documentation

5.39.2.1 [ip_t* tcp_addr_ip](#) (int sockd, [ip_t](#) * output)

[tcp.c](#)

Definition at line 163 of file [tcp.c](#).

References [ip_t::family](#), [ip_t::ip4](#), [ip_t::ip6](#), [MEMORYBUF](#), [mm_alloc\(\)](#), and [mm_copy\(\)](#).

Referenced by [con_init\(\)](#), and [tcp_addr_st\(\)](#).

5.39.2.2 [stringer_t* tcp_addr_st](#) (int sockd, [stringer_t](#) * output)

Definition at line 192 of file [tcp.c](#).

References [ip_presentation\(\)](#), and [tcp_addr_ip\(\)](#).

5.39.2.3 `int tcp_continue (int sockd, int result, int syserror)`

Return -1 if the connection is invalid, 0 if the operation should be retried, or a positive number indicating the number of bytes processed.

Definition at line 84 of file tcp.c.

References `errno_name()`, `log_pedantic`, `MEMORYBUF`, and `status`.

Referenced by `tcp_read()`, and `tcp_write()`.

5.39.2.4 `int_t tcp_error (int error)`

Determine whether the error is a permanent/fatal failure, or a transient error.

Returns:

return 0 for errors we should ignore, and -1 for permanent/fatal failures.

Definition at line 15 of file tcp.c.

References `log_pedantic`, and `MEMORYBUF`.

Referenced by `tcp_status()`.

5.39.2.5 `int tcp_read (int sockd, void * buffer, int length, bool_t block)`

Read data from a TCP/IP network socket.

Parameters:

sockd the socket file descriptor we'll read the data from.

buffer a pointer to the buffer where the data will be stored.

length the maximum length, in bytes, we can read into the buffer.

block a boolean to indicating whether to make a blocking write call.

Returns:

-1 if the network connection is no longer viable, otherwise the number bytes read, while a 0 may indicate the connection wasn't ready (when using non-blocking read calls), or a non-network error may have occurred.

Definition at line 109 of file tcp.c.

References `log_pedantic`, and `tcp_continue()`.

Referenced by `con_read()`, and `con_read_line()`.

5.39.2.6 `int_t tcp_status (int sockd)`

Determine whether the socket connection provided by `sockd` is still valid.

Parameters:

sockd The socket connection being checked.

Returns:

return 0 for valid connections

Definition at line 42 of file tcp.c.

References `MANAGEDBUF`, `st_data_get()`, and `tcp_error()`.

Referenced by `client_read()`, `client_read_line()`, `client_status()`, `client_write()`, and `con_status()`.

5.39.2.7 int tcp_wait (int *sockd*)

Blocks until a socket is ready for a read/write operation, or otherwise becomes invalid.

Parameters:

sockd

Returns:

Definition at line 75 of file tcp.c.

5.39.2.8 int tcp_write (int *sockd*, const void * *buffer*, int *length*, bool_t *block*)

Write data to an open TCP/IP network socket.

Parameters:

sockd the socket file descriptor we'll write the data to.

buffer a pointer to the buffer containing the data to be written.

length the length, in bytes, of the data to be written.

block a boolean to indicating whether to make a blocking write call.

Returns:

-1 on error, or the number of bytes written to the network connection.

Definition at line 140 of file tcp.c.

References `log_pedantic`, and `tcp_continue()`.

Referenced by `con_write_bl()`.

5.40 src/core/indexes/cursors.c File Reference

```
#include "magma.h"
```

Functions

- void [inx_cursor_reset](#) ([inx_cursor_t](#) *cursor)
Reset the position of an inx cursor.
- void [inx_cursor_free](#) ([inx_cursor_t](#) *cursor)
Free an inx cursor and the inx object it points to, if the inx reference count hits zero.
- [inx_cursor_t](#) * [inx_cursor_alloc](#) ([inx_t](#) *index)
Create a new cursor to iterate through an inx object.
- [multi_t](#) [inx_cursor_key_next](#) ([inx_cursor_t](#) *cursor)
Get the key at the next inx cursor position.
- [multi_t](#) [inx_cursor_key_active](#) ([inx_cursor_t](#) *cursor)
Get the key at the current inx cursor position.
- void * [inx_cursor_value_next](#) ([inx_cursor_t](#) *cursor)
Get the key at the next inx cursor position.
- void * [inx_cursor_value_active](#) ([inx_cursor_t](#) *cursor)
Get the value at the current inx cursor position.

5.40.1 Function Documentation

5.40.1.1 [inx_cursor_t](#)* [inx_cursor_alloc](#) ([inx_t](#) * *index*)

Create a new cursor to iterate through an inx object. [cursors.c](#)

Parameters:

index a pointer to the inx object to be traversed.

Returns:

NULL on failure, or a pointer to a new inx cursor for the specified inx object on success.

Definition at line 52 of file [cursors.c](#).

References [inx_auto_unlock\(\)](#), [inx_auto_write\(\)](#), and [inx_cursor_t](#).

Referenced by [config_load_cmdline_settings\(\)](#), [config_load_file_settings\(\)](#), [contact_find_detail\(\)](#), [contact_find_name\(\)](#), [contacts_fetch\(\)](#), [contacts_update\(\)](#), [http_data_get\(\)](#), [imap_append\(\)](#), [imap_close\(\)](#), [imap_copy\(\)](#), [imap_duplicate_messages\(\)](#), [imap_examine\(\)](#), [imap_expunge\(\)](#), [imap_fetch\(\)](#), [imap_folder_remove\(\)](#), [imap_folder_status\(\)](#), [imap_list\(\)](#), [imap_lsub\(\)](#), [imap_narrow_folders\(\)](#), [imap_narrow_messages\(\)](#), [imap_next_folder_order\(\)](#), [imap_search\(\)](#), [imap_search_messages\(\)](#), [imap_select\(\)](#), [imap_session_destroy\(\)](#), [imap_session_update\(\)](#), [imap_store\(\)](#), [imap_update_flags\(\)](#), [magma_folder_children\(\)](#), [magma_folder_find_full_name\(\)](#), [magma_folder_find_name\(\)](#), [messages_update\(\)](#), [meta_data_fetch_messages\(\)](#), [meta_data_flags_add\(\)](#), [meta_data_flags_remove\(\)](#), [meta_data_flags_replace\(\)](#), [meta_folders_by_name\(\)](#), [meta_folders_by_number\(\)](#), [meta_folders_children\(\)](#), [meta_folders_stats_tags\(\)](#), [meta_message_by_number\(\)](#), [meta_messages_update_sequences\(\)](#), [obj_cache_prune\(\)](#), [pattern_check\(\)](#), [pop_get_last\(\)](#), [pop_get_message\(\)](#), [pop_list\(\)](#),

pop_session_destroy(), pop_session_reset(), pop_total_messages(), pop_total_size(), pop_uidl(), portal_config_collection(), portal_contact_details(), portal_endpoint_alert_list(), portal_endpoint_aliases(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_list(), portal_endpoint_folders_list(), portal_endpoint_folders_tags(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_tag(), portal_endpoint_messages_tags(), portal_smtp_create_data(), portal_smtp_merge_headers(), portal_smtp_relay_message(), portal_upload(), register_abuse_check_blocklist(), signature_tree_get(), signature_tree_verify(), smtp_bypass_check(), smtp_check_filters(), and teacher_print_message().

5.40.1.2 void `inx_cursor_free` (`inx_cursor_t * cursor`)

Free an inx cursor and the inx object it points to, if the inx reference count hits zero.

Parameters:

cursor the inx cursor to be freed.

Returns:

This function returns no value.

Definition at line 29 of file `cursors.c`.

References `inx_auto_unlock()`, `inx_auto_write()`, `inx_free()`, and `inx_t`.

Referenced by `config_load_cmdline_settings()`, `config_load_file_settings()`, `contact_find_detail()`, `contact_find_name()`, `contacts_fetch()`, `contacts_update()`, `http_data_get()`, `imap_append()`, `imap_close()`, `imap_copy()`, `imap_duplicate_messages()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_folder_remove()`, `imap_folder_status()`, `imap_list()`, `imap_lsub()`, `imap_narrow_folders()`, `imap_narrow_messages()`, `imap_next_folder_order()`, `imap_search()`, `imap_search_messages()`, `imap_select()`, `imap_session_destroy()`, `imap_session_update()`, `imap_store()`, `imap_update_flags()`, `magma_folder_children()`, `magma_folder_find_full_name()`, `magma_folder_find_name()`, `messages_update()`, `meta_data_fetch_messages()`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_flags_replace()`, `meta_folders_by_name()`, `meta_folders_by_number()`, `meta_folders_children()`, `meta_folders_stats_tags()`, `meta_message_by_number()`, `meta_messages_update_sequences()`, `obj_cache_prune()`, `pattern_check()`, `pop_get_last()`, `pop_get_message()`, `pop_list()`, `pop_session_destroy()`, `pop_session_reset()`, `pop_total_messages()`, `pop_total_size()`, `pop_uidl()`, `portal_config_collection()`, `portal_contact_details()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_list()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_smtp_create_data()`, `portal_smtp_merge_headers()`, `portal_smtp_relay_message()`, `portal_upload()`, `register_abuse_check_blocklist()`, `signature_tree_get()`, `signature_tree_verify()`, `smtp_bypass_check()`, `smtp_check_filters()`, and `teacher_print_message()`.

5.40.1.3 multi_t `inx_cursor_key_active` (`inx_cursor_t * cursor`)

Get the key at the current inx cursor position.

Parameters:

cursor the inx cursor to be examined.

Returns:

NULL on failure, or a multi-type data object containing the key of the specified inx cursor.

Definition at line 92 of file `cursors.c`.

References `inx_auto_read()`, `inx_auto_unlock()`, and `mt_get_null()`.

Referenced by `imap_search_messages()`, and `obj_cache_prune()`.

5.40.1.4 multi_t `inx_cursor_key_next` (`inx_cursor_t * cursor`)

Get the key at the next inx cursor position.

Parameters:

cursor the inx cursor to be examined.

Returns:

NULL on failure, or a multi-type data object containing the key of the next inx cursor position.

Definition at line 74 of file cursors.c.

References `inx_auto_read()`, `inx_auto_unlock()`, and `mt_get_null()`.

Referenced by `config_load_cmdline_settings()`, and `config_load_file_settings()`.

5.40.1.5 void inx_cursor_reset (inx_cursor_t * cursor)

Reset the position of an inx cursor.

Parameters:

cursor a pointer to the inx cursor to be reset.

Returns:

This function returns no value.

Definition at line 15 of file cursors.c.

Referenced by `imap_fetch()`, `imap_list()`, `imap_lsub()`, `obj_cache_prune()`, and `portal_smtp_create_data()`.

5.40.1.6 void* inx_cursor_value_active (inx_cursor_t * cursor)

Get the value at the current inx cursor position.

Parameters:

cursor the inx cursor to be examined.

Returns:

NULL on failure, or the value at the current position of the specified inx cursor.

Definition at line 128 of file cursors.c.

References `inx_auto_read()`, and `inx_auto_unlock()`.

Referenced by `config_load_cmdline_settings()`, and `config_load_file_settings()`.

5.40.1.7 void* inx_cursor_value_next (inx_cursor_t * cursor)

Get the key at the next inx cursor position.

Parameters:

cursor the inx cursor to be examined.

Returns:

NULL on failure, or a multi-type data object containing the key of the next inx cursor position.

Definition at line 110 of file cursors.c.

References `inx_auto_read()`, and `inx_auto_unlock()`.

Referenced by `contact_find_detail()`, `contact_find_name()`, `contacts_fetch()`, `contacts_update()`, `http_data_get()`, `imap_append()`, `imap_close()`, `imap_copy()`, `imap_duplicate_messages()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_folder_remove()`, `imap_folder_status()`, `imap_list()`, `imap_lsub()`, `imap_narrow_folders()`, `imap_narrow_messages()`, `imap_next_folder_order()`, `imap_search()`, `imap_search_messages()`, `imap_select()`, `imap_session_destroy()`, `imap_session_update()`, `imap_store()`, `imap_update_flags()`, `magma_folder_children()`, `magma_folder_find_full_name()`, `magma_folder_find_name()`, `messages_update()`, `meta_data_fetch_messages()`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_flags_replace()`, `meta_folders_by_name()`, `meta_folders_by_number()`, `meta_folders_children()`, `meta_folders_stats_tags()`, `meta_message_by_number()`, `meta_messages_update_sequences()`, `obj_cache_prune()`, `pattern_check()`, `pop_get_last()`, `pop_get_message()`, `pop_list()`, `pop_session_destroy()`, `pop_session_reset()`, `pop_total_messages()`, `pop_total_size()`, `pop_uidl()`, `portal_config_collection()`, `portal_contact_details()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_list()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_smtp_create_data()`, `portal_smtp_merge_headers()`, `portal_smtp_relay_message()`, `portal_upload()`, `register_abuse_check_blocklist()`, `signature_tree_get()`, `signature_tree_verify()`, `smtp_bypass_check()`, `smtp_check_filters()`, and `teacher_print_message()`.

5.41 src/core/indexes/hashed.c File Reference

```
#include "magma.h"
```

Defines

- #define [MAGMA_HASHED_BUCKETS](#) 1024

Functions

- struct [__attribute__](#) ((packed))
- uint32_t [hashed_bucket](#) (uint32_t buckets, [multi_t](#) key)
Get the hash bucket for a key.
- [hashed_bucket_t](#) * [hashed_bucket_alloc](#) ([multi_t](#) key, void *data)
Allocate and initialize hashed bucket object.
- [hashed_bucket_t](#) * [hashed_bucket_get_ptr](#) ([hashed_index_t](#) *hashed, uint32_t num)
- void [hashed_bucket_set_ptr](#) ([hashed_index_t](#) *hashed, uint32_t num, [hashed_bucket_t](#) *bucket)
- void [hashed_bucket_add](#) ([hashed_bucket_t](#) *bucket, [hashed_bucket_t](#) *new)
- [bool_t](#) [hashed_insert](#) (void *inx, [multi_t](#) key, void *data)
- void * [hashed_bucket_find_key](#) ([hashed_bucket_t](#) *bucket, [multi_t](#) key)
- void * [hashed_find](#) (void *inx, [multi_t](#) key)
- [bool_t](#) [hashed_delete](#) (void *inx, [multi_t](#) key)
- [hashed_bucket_t](#) * [hashed_cursor_active](#) ([hashed_cursor_t](#) *cursor)
- [hashed_bucket_t](#) * [hashed_cursor_next](#) ([hashed_cursor_t](#) *cursor)
- void * [hashed_cursor_value_next](#) ([hashed_cursor_t](#) *cursor)
- void * [hashed_cursor_value_active](#) ([hashed_cursor_t](#) *cursor)
- [multi_t](#) [hashed_cursor_key_next](#) ([hashed_cursor_t](#) *cursor)
- [multi_t](#) [hashed_cursor_key_active](#) ([hashed_cursor_t](#) *cursor)
- void [hashed_cursor_reset](#) ([hashed_cursor_t](#) *cursor)
- void [hashed_cursor_free](#) ([hashed_cursor_t](#) *cursor)
- void * [hashed_cursor_alloc](#) ([inx_t](#) *inx)
- void [hashed_free](#) (void *inx)
- void [hashed_truncate](#) (void *inx)
- [inx_t](#) * [hashed_alloc](#) (uint64_t options, void *data_free)
Allocate a new hash table.

Variables

- [hashed_bucket_t](#)
- [hashed_cursor_t](#)
- [hashed_index_t](#)

5.41.1 Define Documentation

5.41.1.1 #define MAGMA_HASHED_BUCKETS 1024

Definition at line 10 of file hashed.c.

Referenced by [hashed_alloc\(\)](#).

5.41.2 Function Documentation

5.41.2.1 struct __attribute__((packed)) [read]

Definition at line 25 of file hashed.c.

5.41.2.2 inx_t* hashed_alloc (uint64_t options, void * data_free)

Allocate a new hash table. [hashed.c](#)

Parameters:

options an options value for the hash table.

data_free a pointer to a function

Returns:

NULL on failure, or a pointer to the newly allocated hash table object on success.

Definition at line 496 of file hashed.c.

References `hashed_bucket_t`, `hashed_cursor_alloc()`, `hashed_cursor_free()`, `hashed_cursor_key_active()`, `hashed_cursor_key_next()`, `hashed_cursor_reset()`, `hashed_cursor_value_active()`, `hashed_cursor_value_next()`, `hashed_delete()`, `hashed_find()`, `hashed_free()`, `hashed_index_t`, `hashed_insert()`, `hashed_truncate()`, `inx_t`, `MAGMA_HASHED_BUCKETS`, `mm_alloc()`, and `mm_free()`.

Referenced by `inx_alloc()`.

5.41.2.3 uint32_t hashed_bucket (uint32_t buckets, multi_t key)

Get the hash bucket for a key.

Note:

If the key is passed as a string, it will be mapped to a bucket using the Fletcher32 hash.

Parameters:

buckets the total number of buckets for grouping items.

key a multi-type key with the value to be looked up; numbers and strings are supported.

Returns:

the number of the hash bucket corresponding to the specified key.

Definition at line 40 of file hashed.c.

References `bitwise_count()`, `count`, `hash_fletcher32()`, `log_check`, `mt_get_char()`, `mt_get_length()`, `mt_get_number()`, and `mt_is_number()`.

Referenced by `hashed_delete()`, `hashed_find()`, and `hashed_insert()`.

5.41.2.4 void hashed_bucket_add (hashed_bucket_t * bucket, hashed_bucket_t * new)

Definition at line 119 of file hashed.c.

References `hashed_bucket_t`.

Referenced by `hashed_insert()`.

5.41.2.5 hashed_bucket_t* hashed_bucket_alloc (multi_t *key*, void * *data*)

Allocate and initialize hashed bucket object.

Parameters:

key a multi-type key that will be hashed on lookup.

data a pointer to a data buffer associated with the key.

Returns:

NULL on failure, or a pointer to the newly allocated hashed bucket object on success.

Definition at line 70 of file hashed.c.

References hashed_bucket_t, mm_alloc(), and mt_dupe().

Referenced by hashed_insert().

5.41.2.6 void* hashed_bucket_find_key (hashed_bucket_t * *bucket*, multi_t *key*)

Definition at line 173 of file hashed.c.

References data, hashed_bucket_t, and ident_mt_mt().

Referenced by hashed_find().

5.41.2.7 hashed_bucket_t* hashed_bucket_get_ptr (hashed_index_t * *hashed*, uint32_t *num*)

Definition at line 85 of file hashed.c.

References hashed_bucket_t, hashed_index_t, and log_pedantic.

Referenced by hashed_cursor_active(), hashed_delete(), hashed_find(), hashed_free(), hashed_insert(), and hashed_truncate().

5.41.2.8 void hashed_bucket_set_ptr (hashed_index_t * *hashed*, uint32_t *num*, hashed_bucket_t * *bucket*)

Definition at line 103 of file hashed.c.

References hashed_bucket_t, hashed_index_t, and log_pedantic.

Referenced by hashed_delete(), hashed_insert(), and hashed_truncate().

5.41.2.9 hashed_bucket_t* hashed_cursor_active (hashed_cursor_t * *cursor*)

BUG: It's a little unclear what's happening, but when this function gets called, the serials don't match, and the count is greater than the number of available data buckets, because a bucket has been removed, the cursor will get stuck returning the last item endlessly (I think).

Definition at line 274 of file hashed.c.

References count, hashed_bucket_get_ptr(), hashed_bucket_t, and hashed_index_t.

Referenced by hashed_cursor_key_active(), hashed_cursor_next(), and hashed_cursor_value_active().

5.41.2.10 void* hashed_cursor_alloc (inx_t * *inx*)

Definition at line 406 of file hashed.c.

References hashed_cursor_t, log_pedantic, and mm_alloc().

Referenced by hashed_alloc().

5.41.2.11 void hashed_cursor_free (hashed_cursor_t * *cursor*)

Definition at line 397 of file hashed.c.

References mm_free().

Referenced by hashed_alloc().

5.41.2.12 multi_t hashed_cursor_key_active (hashed_cursor_t * *cursor*)

Definition at line 377 of file hashed.c.

References hashed_bucket_t, hashed_cursor_active(), and mt_get_null().

Referenced by hashed_alloc().

5.41.2.13 multi_t hashed_cursor_key_next (hashed_cursor_t * *cursor*)

Definition at line 367 of file hashed.c.

References hashed_bucket_t, hashed_cursor_next(), and mt_get_null().

Referenced by hashed_alloc().

5.41.2.14 hashed_bucket_t* hashed_cursor_next (hashed_cursor_t * *cursor*)

Definition at line 322 of file hashed.c.

References hashed_bucket_t, hashed_cursor_active(), and hashed_index_t.

Referenced by hashed_cursor_key_next(), and hashed_cursor_value_next().

5.41.2.15 void hashed_cursor_reset (hashed_cursor_t * *cursor*)

Definition at line 387 of file hashed.c.

Referenced by hashed_alloc().

5.41.2.16 void* hashed_cursor_value_active (hashed_cursor_t * *cursor*)

Definition at line 357 of file hashed.c.

References hashed_bucket_t, and hashed_cursor_active().

Referenced by hashed_alloc().

5.41.2.17 void* hashed_cursor_value_next (hashed_cursor_t * *cursor*)

Definition at line 346 of file hashed.c.

References hashed_bucket_t, and hashed_cursor_next().

Referenced by hashed_alloc().

5.41.2.18 bool_t hashed_delete (void * *inx*, multi_t *key*)

Definition at line 216 of file hashed.c.

References `hashed_bucket()`, `hashed_bucket_get_ptr()`, `hashed_bucket_set_ptr()`, `hashed_bucket_t`, `hashed_index_t`, `ident_mt_mt()`, `inx_t`, `mm_free()`, and `mt_free()`.

Referenced by `hashed_alloc()`.

5.41.2.19 `void* hashed_find (void * inx, multi_t key)`

Definition at line 188 of file `hashed.c`.

References `data`, `hashed_bucket()`, `hashed_bucket_find_key()`, `hashed_bucket_get_ptr()`, `hashed_bucket_t`, `hashed_index_t`, and `inx_t`.

Referenced by `hashed_alloc()`.

5.41.2.20 `void hashed_free (void * inx)`

Definition at line 420 of file `hashed.c`.

References `hashed_bucket_get_ptr()`, `hashed_bucket_t`, `hashed_index_t`, `inx_t`, `mm_free()`, and `mt_free()`.

Referenced by `hashed_alloc()`.

5.41.2.21 `bool_t hashed_insert (void * inx, multi_t key, void * data)`

Definition at line 136 of file `hashed.c`.

References `hashed_bucket()`, `hashed_bucket_add()`, `hashed_bucket_alloc()`, `hashed_bucket_get_ptr()`, `hashed_bucket_set_ptr()`, `hashed_bucket_t`, `hashed_index_t`, and `inx_t`.

Referenced by `hashed_alloc()`.

5.41.2.22 `void hashed_truncate (void * inx)`

Definition at line 453 of file `hashed.c`.

References `hashed_bucket_get_ptr()`, `hashed_bucket_set_ptr()`, `hashed_bucket_t`, `hashed_index_t`, `inx_t`, `mm_free()`, and `mt_free()`.

Referenced by `hashed_alloc()`.

5.41.3 Variable Documentation

5.41.3.1 `hashed_bucket_t`

Definition at line 17 of file `hashed.c`.

Referenced by `hashed_alloc()`, `hashed_bucket_add()`, `hashed_bucket_alloc()`, `hashed_bucket_find_key()`, `hashed_bucket_get_ptr()`, `hashed_bucket_set_ptr()`, `hashed_cursor_active()`, `hashed_cursor_key_active()`, `hashed_cursor_key_next()`, `hashed_cursor_next()`, `hashed_cursor_value_active()`, `hashed_cursor_value_next()`, `hashed_delete()`, `hashed_find()`, `hashed_free()`, `hashed_insert()`, and `hashed_truncate()`.

5.41.3.2 `hashed_cursor_t`

Definition at line 23 of file `hashed.c`.

Referenced by `hashed_cursor_alloc()`.

5.41.3.3 `hashed_index_t`

Definition at line 27 of file `hashed.c`.

Referenced by hashed_alloc(), hashed_bucket_get_ptr(), hashed_bucket_set_ptr(), hashed_cursor_active(), hashed_cursor_next(), hashed_delete(), hashed_find(), hashed_free(), hashed_insert(), and hashed_truncate().

5.42 src/core/indexes/indexes.h File Reference

Defines

- #define [MAGMA_INDEX_TYPE](#) (M_INX_TREE | M_INX_LINKED | M_INX_HASHED)
- #define [MAGMA_INDEX_OPTION](#) (M_INX_INDEX_LOCK)

Enumerations

- enum [MAGMA_INDEX](#) { [M_INX_TREE](#) = 1, [M_INX_HASHED](#) = 2, [M_INX_LINKED](#) = 4, [M_INX_LOCK_MANUAL](#) = 16 }

Functions

- struct [__attribute__](#) ((packed))
- [inx_cursor_t](#) * [inx_cursor_alloc](#) ([inx_t](#) *index)
cursors.c
- void [inx_cursor_free](#) ([inx_cursor_t](#) *cursor)
Free an inx cursor and the inx object it points to, if the inx reference count hits zero.
- [multi_t](#) [inx_cursor_key_active](#) ([inx_cursor_t](#) *cursor)
Get the key at the current inx cursor position.
- [multi_t](#) [inx_cursor_key_next](#) ([inx_cursor_t](#) *cursor)
Get the key at the next inx cursor position.
- void [inx_cursor_reset](#) ([inx_cursor_t](#) *cursor)
Reset the position of an inx cursor.
- void * [inx_cursor_value_active](#) ([inx_cursor_t](#) *cursor)
Get the value at the current inx cursor position.
- void * [inx_cursor_value_next](#) ([inx_cursor_t](#) *cursor)
Get the key at the next inx cursor position.
- [inx_t](#) * [inx_alloc](#) (uint64_t options, void *data_free)
inx.c
- [bool_t](#) [inx_append](#) ([inx_t](#) *inx, [multi_t](#) key, void *data)
Append a new record onto an inx holder.
- void [inx_auto_read](#) ([inx_t](#) *inx)
- void [inx_auto_unlock](#) ([inx_t](#) *inx)
- void [inx_auto_write](#) ([inx_t](#) *inx)
- void [inx_cleanup](#) ([inx_t](#) *inx)
Perform a checked free of an inx object.
- uint64_t [inx_count](#) ([inx_t](#) *inx)
Return the total number of items held by an inx object.
- [bool_t](#) [inx_delete](#) ([inx_t](#) *inx, [multi_t](#) key)
Delete a child of an inx object with a specified key.

- void * [inx_find](#) ([inx_t](#) *inx, [multi_t](#) key)

Find the value associated with a particular key within the children of an inx object.

- void [inx_free](#) ([inx_t](#) *inx)
- [bool_t](#) [inx_insert](#) ([inx_t](#) *inx, [multi_t](#) key, void *data)

Insert a new record into an inx holder.

- void [inx_lock_read](#) ([inx_t](#) *inx)

Acquire a reader's lock for an inx object.

- void [inx_lock_write](#) ([inx_t](#) *inx)

Acquire a writer's lock for an inx object.

- [uint64_t](#) [inx_options](#) ([inx_t](#) *inx)

Return the options value of an inx object.

- [bool_t](#) [inx_replace](#) ([inx_t](#) *inx, [multi_t](#) key, void *data)

Replace the value of a key in an inx holder.

- [uint64_t](#) [inx_serial](#) ([inx_t](#) *inx)

- void [inx_truncate](#) ([inx_t](#) *inx)

- void [inx_unlock](#) ([inx_t](#) *inx)

Unlock an inx object.

- [inx_t](#) * [linked_alloc](#) ([uint64_t](#) options, void *data_free)

[linked.c](#)

- [inx_t](#) * [hashed_alloc](#) ([uint64_t](#) options, void *data_free)

[hashed.c](#)

Variables

- [inx_t](#)
- [inx_cursor_t](#)

5.42.1 Define Documentation

5.42.1.1 #define MAGMA_INDEX_OPTION (M_INX_INDEX_LOCK)

The different index options.

Definition at line 32 of file indexes.h.

5.42.1.2 #define MAGMA_INDEX_TYPE (M_INX_TREE | M_INX_LINKED | M_INX_HASHED)

The different types of indexes.

Definition at line 27 of file indexes.h.

Referenced by [inx_alloc\(\)](#).

5.42.2 Enumeration Type Documentation

5.42.2.1 enum MAGMA_INDEX

Index types and options.

Enumerator:

M_INX_TREE M_INX_BTREE.
M_INX_HASHED M_INX_HASHED.
M_INX_LINKED M_INX_LINKED.
M_INX_LOCK_MANUAL M_INX_LOCK_MANUAL.

Definition at line 15 of file indexes.h.

5.42.3 Function Documentation

5.42.3.1 struct __attribute__((packed)) [read]

Definition at line 64 of file indexes.h.

References `inx_t`.

5.42.3.2 inx_t* hashed_alloc (uint64_t options, void * data_free)

[hashed.c](#) [hashed.c](#)

Parameters:

options an options value for the hash table.
data_free a pointer to a function

Returns:

NULL on failure, or a pointer to the newly allocated hash table object on success.

Definition at line 496 of file hashed.c.

References `hashed_bucket_t`, `hashed_cursor_alloc()`, `hashed_cursor_free()`, `hashed_cursor_key_active()`, `hashed_cursor_key_next()`, `hashed_cursor_reset()`, `hashed_cursor_value_active()`, `hashed_cursor_value_next()`, `hashed_delete()`, `hashed_find()`, `hashed_free()`, `hashed_index_t`, `hashed_insert()`, `hashed_truncate()`, `inx_t`, `MAGMA_HASHED_BUCKETS`, `mm_alloc()`, and `mm_free()`.

Referenced by `inx_alloc()`.

5.42.3.3 inx_t* inx_alloc (uint64_t options, void * data_free)

[inx.c](#) [inx.c](#)

Parameters:

options a value indicating the inx type. Can be `M_INX_TREE` for a binary tree, `M_INX_LINKED` for a linked list, or `M_INX_HASHED` for a hash tree.
data_free a function pointer to a routine to free the data associated with an inx record.

Returns:

NULL on failure or a pointer to the newly created inx object on success.

Definition at line 319 of file `inx.c`.

References `hashed_alloc()`, `inx_t`, `linked_alloc()`, `log_options`, `M_INX_HASHED`, `M_INX_LINKED`, `M_INX_LOCK_MANUAL`, `M_INX_TREE`, `M_LOG_ERROR`, `M_LOG_STACK_TRACE`, `MAGMA_INDEX_TYPE`, `rwlock_init()`, and `tree_alloc()`.

Referenced by `contact_alloc()`, `contact_folder_alloc()`, `http_content_refresh()`, `http_content_start()`, `http_parse_header()`, `http_parse_pairs()`, `imap_duplicate_messages()`, `imap_narrow_folders()`, `imap_narrow_messages()`, `imap_search_messages()`, `magma_folder_fetch()`, `message_folder_alloc()`, `meta_data_fetch_alerts()`, `meta_data_fetch_all_tags()`, `meta_data_fetch_folders()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_messages()`, `meta_folders_stats_tags()`, `nvp_alloc()`, `obj_cache_start()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_tag()`, `portal_parse_json_str_array()`, `register_data_fetch_blocklist()`, `sess_create()`, `signature_tree_alloc()`, `smtp_add_bypass_entry()`, `smtp_fetch_inbound()`, `teacher_add_cookie()`, `user_config_alloc()`, `warehouse_fetch_domains()`, and `warehouse_fetch_patterns()`.

5.42.3.4 `bool_t inx_append (inx_t * inx, multi_t key, void * data)`

Append a new record onto an `inx` holder.

Note:

This function only provides benefits for some `inx` types (like linked lists). For other index types, it simply functions as an alias for the `insert` function.

Parameters:

inx a pointer to the `inx` object that will hold the record.

key a multi-type value specifying the identifier of the record to be inserted.

data a pointer to the data associated with the new key.

Returns:

true if the new record was inserted successfully or false on failure.

Definition at line 140 of file `inx.c`.

References `inx_auto_unlock()`, `inx_auto_write()`, and `log_pedantic`.

Referenced by `imap_append_message()`, `imap_duplicate_messages()`, `imap_message_copier()`, `imap_narrow_messages()`, `imap_search_messages()`, `meta_data_fetch_folder_messages()`, and `meta_data_fetch_messages()`.

5.42.3.5 `void inx_auto_read (inx_t * inx)`

Definition at line 41 of file `inx.c`.

References `rwlock_lock_read()`.

Referenced by `inx_count()`, `inx_cursor_key_active()`, `inx_cursor_key_next()`, `inx_cursor_value_active()`, `inx_cursor_value_next()`, `inx_find()`, `inx_options()`, and `inx_serial()`.

5.42.3.6 `void inx_auto_unlock (inx_t * inx)`

Definition at line 22 of file `inx.c`.

References `rwlock_unlock()`.

Referenced by `inx_append()`, `inx_count()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_key_active()`, `inx_cursor_key_next()`, `inx_cursor_value_active()`, `inx_cursor_value_next()`, `inx_delete()`, `inx_find()`, `inx_free()`, `inx_insert()`, `inx_options()`, `inx_replace()`, `inx_serial()`, and `inx_truncate()`.

5.42.3.7 void `inx_auto_write` (`inx_t *inx`)

Definition at line 60 of file `inx.c`.

References `rwlock_lock_write()`.

Referenced by `inx_append()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_delete()`, `inx_free()`, `inx_insert()`, `inx_replace()`, and `inx_truncate()`.

5.42.3.8 void `inx_cleanup` (`inx_t *inx`)

Perform a checked free of an `inx` object.

See also:

[inx_free\(\)](#)

Parameters:

inx the `inx` instance to be freed.

Definition at line 306 of file `inx.c`.

References `inx_free()`.

Referenced by `contact_free()`, `contacts_update()`, `domain_stop()`, `http_content_stop()`, `http_session_reset()`, `imap_narrow_messages()`, `magma_folder_free()`, `messages_update()`, `meta_data_fetch_folders()`, `meta_data_fetch_mailbox_aliases()`, `meta_free()`, `pattern_stop()`, `pattern_update()`, `sess_destroy()`, `sess_release_composition()`, `smtp_free_inbound()`, and `user_config_free()`.

5.42.3.9 uint64_t `inx_count` (`inx_t *inx`)

Return the total number of items held by an `inx` object.

Parameters:

inx a pointer to the `inx` object to be examined.

Returns:

the total number of items held by the `inx` object.

Definition at line 95 of file `inx.c`.

References `count`, `inx_auto_read()`, `inx_auto_unlock()`, and `log_pedantic`.

Referenced by `contact_folder_remove()`, `imap_copy()`, `imap_login()`, `imap_narrow_messages()`, `message_folder_remove()`, `obj_cache_prune()`, `pop_pass()`, `signature_tree_add()`, `signature_tree_get()`, and `signature_tree_verify()`.

5.42.3.10 inx_cursor_t* `inx_cursor_alloc` (`inx_t *index`)

[cursors.c](#) [cursors.c](#)

Parameters:

index a pointer to the `inx` object to be traversed.

Returns:

NULL on failure, or a pointer to a new `inx` cursor for the specified `inx` object on success.

Definition at line 52 of file cursors.c.

References `inx_auto_unlock()`, `inx_auto_write()`, and `inx_cursor_t`.

Referenced by `config_load_cmdline_settings()`, `config_load_file_settings()`, `contact_find_detail()`, `contact_find_name()`, `contacts_fetch()`, `contacts_update()`, `http_data_get()`, `imap_append()`, `imap_close()`, `imap_copy()`, `imap_duplicate_messages()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_folder_remove()`, `imap_folder_status()`, `imap_list()`, `imap_lsub()`, `imap_narrow_folders()`, `imap_narrow_messages()`, `imap_next_folder_order()`, `imap_search()`, `imap_search_messages()`, `imap_select()`, `imap_session_destroy()`, `imap_session_update()`, `imap_store()`, `imap_update_flags()`, `magma_folder_children()`, `magma_folder_find_full_name()`, `magma_folder_find_name()`, `messages_update()`, `meta_data_fetch_messages()`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_flags_replace()`, `meta_folders_by_name()`, `meta_folders_by_number()`, `meta_folders_children()`, `meta_folders_stats_tags()`, `meta_message_by_number()`, `meta_messages_update_sequences()`, `obj_cache_prune()`, `pattern_check()`, `pop_get_last()`, `pop_get_message()`, `pop_list()`, `pop_session_destroy()`, `pop_session_reset()`, `pop_total_messages()`, `pop_total_size()`, `pop_uidl()`, `portal_config_collection()`, `portal_contact_details()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_list()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_smtp_create_data()`, `portal_smtp_merge_headers()`, `portal_smtp_relay_message()`, `portal_upload()`, `register_abuse_check_blocklist()`, `signature_tree_get()`, `signature_tree_verify()`, `smtp_bypass_check()`, `smtp_check_filters()`, and `teacher_print_message()`.

5.42.3.11 void inx_cursor_free (inx_cursor_t * cursor)

Free an inx cursor and the inx object it points to, if the inx reference count hits zero.

Parameters:

cursor the inx cursor to be freed.

Returns:

This function returns no value.

Definition at line 29 of file cursors.c.

References `inx_auto_unlock()`, `inx_auto_write()`, `inx_free()`, and `inx_t`.

Referenced by `config_load_cmdline_settings()`, `config_load_file_settings()`, `contact_find_detail()`, `contact_find_name()`, `contacts_fetch()`, `contacts_update()`, `http_data_get()`, `imap_append()`, `imap_close()`, `imap_copy()`, `imap_duplicate_messages()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_folder_remove()`, `imap_folder_status()`, `imap_list()`, `imap_lsub()`, `imap_narrow_folders()`, `imap_narrow_messages()`, `imap_next_folder_order()`, `imap_search()`, `imap_search_messages()`, `imap_select()`, `imap_session_destroy()`, `imap_session_update()`, `imap_store()`, `imap_update_flags()`, `magma_folder_children()`, `magma_folder_find_full_name()`, `magma_folder_find_name()`, `messages_update()`, `meta_data_fetch_messages()`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_flags_replace()`, `meta_folders_by_name()`, `meta_folders_by_number()`, `meta_folders_children()`, `meta_folders_stats_tags()`, `meta_message_by_number()`, `meta_messages_update_sequences()`, `obj_cache_prune()`, `pattern_check()`, `pop_get_last()`, `pop_get_message()`, `pop_list()`, `pop_session_destroy()`, `pop_session_reset()`, `pop_total_messages()`, `pop_total_size()`, `pop_uidl()`, `portal_config_collection()`, `portal_contact_details()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_list()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_smtp_create_data()`, `portal_smtp_merge_headers()`, `portal_smtp_relay_message()`, `portal_upload()`, `register_abuse_check_blocklist()`, `signature_tree_get()`, `signature_tree_verify()`, `smtp_bypass_check()`, `smtp_check_filters()`, and `teacher_print_message()`.

5.42.3.12 multi_t inx_cursor_key_active (inx_cursor_t * cursor)

Get the key at the current inx cursor position.

Parameters:

cursor the inx cursor to be examined.

Returns:

NULL on failure, or a multi-type data object containing the key of the specified inx cursor.

Definition at line 92 of file cursors.c.

References `inx_auto_read()`, `inx_auto_unlock()`, and `mt_get_null()`.

Referenced by `imap_search_messages()`, and `obj_cache_prune()`.

5.42.3.13 `multi_t inx_cursor_key_next (inx_cursor_t * cursor)`

Get the key at the next inx cursor position.

Parameters:

cursor the inx cursor to be examined.

Returns:

NULL on failure, or a multi-type data object containing the key of the next inx cursor position.

Definition at line 74 of file cursors.c.

References `inx_auto_read()`, `inx_auto_unlock()`, and `mt_get_null()`.

Referenced by `config_load_cmdline_settings()`, and `config_load_file_settings()`.

5.42.3.14 `void inx_cursor_reset (inx_cursor_t * cursor)`

Reset the position of an inx cursor.

Parameters:

cursor a pointer to the inx cursor to be reset.

Returns:

This function returns no value.

Definition at line 15 of file cursors.c.

Referenced by `imap_fetch()`, `imap_list()`, `imap_lsub()`, `obj_cache_prune()`, and `portal_smtp_create_data()`.

5.42.3.15 `void* inx_cursor_value_active (inx_cursor_t * cursor)`

Get the value at the current inx cursor position.

Parameters:

cursor the inx cursor to be examined.

Returns:

NULL on failure, or the value at the current position of the specified inx cursor.

Definition at line 128 of file cursors.c.

References `inx_auto_read()`, and `inx_auto_unlock()`.

Referenced by `config_load_cmdline_settings()`, and `config_load_file_settings()`.

5.42.3.16 void* *inx_cursor_value_next* (*inx_cursor_t* * *cursor*)

Get the key at the next *inx* cursor position.

Parameters:

cursor the *inx* cursor to be examined.

Returns:

NULL on failure, or a multi-type data object containing the key of the next *inx* cursor position.

Definition at line 110 of file *cursors.c*.

References *inx_auto_read()*, and *inx_auto_unlock()*.

Referenced by *contact_find_detail()*, *contact_find_name()*, *contacts_fetch()*, *contacts_update()*, *http_data_get()*, *imap_append()*, *imap_close()*, *imap_copy()*, *imap_duplicate_messages()*, *imap_examine()*, *imap_expunge()*, *imap_fetch()*, *imap_folder_remove()*, *imap_folder_status()*, *imap_list()*, *imap_lsub()*, *imap_narrow_folders()*, *imap_narrow_messages()*, *imap_next_folder_order()*, *imap_search()*, *imap_search_messages()*, *imap_select()*, *imap_session_destroy()*, *imap_session_update()*, *imap_store()*, *imap_update_flags()*, *magma_folder_children()*, *magma_folder_find_full_name()*, *magma_folder_find_name()*, *messages_update()*, *meta_data_fetch_messages()*, *meta_data_flags_add()*, *meta_data_flags_remove()*, *meta_data_flags_replace()*, *meta_folders_by_name()*, *meta_folders_by_number()*, *meta_folders_children()*, *meta_folders_stats_tags()*, *meta_message_by_number()*, *meta_messages_update_sequences()*, *obj_cache_prune()*, *pattern_check()*, *pop_get_last()*, *pop_get_message()*, *pop_list()*, *pop_session_destroy()*, *pop_session_reset()*, *pop_total_messages()*, *pop_total_size()*, *pop_uidl()*, *portal_config_collection()*, *portal_contact_details()*, *portal_endpoint_alert_list()*, *portal_endpoint_aliases()*, *portal_endpoint_contacts_add()*, *portal_endpoint_contacts_copy()*, *portal_endpoint_contacts_list()*, *portal_endpoint_folders_list()*, *portal_endpoint_folders_tags()*, *portal_endpoint_messages_flag()*, *portal_endpoint_messages_list()*, *portal_endpoint_messages_tag()*, *portal_endpoint_messages_tags()*, *portal_smtp_create_data()*, *portal_smtp_merge_headers()*, *portal_smtp_relay_message()*, *portal_upload()*, *register_abuse_check_blocklist()*, *signature_tree_get()*, *signature_tree_verify()*, *smtp_bypass_check()*, *smtp_check_filters()*, and *teacher_print_message()*.

5.42.3.17 bool_t *inx_delete* (*inx_t* * *inx*, multi_t *key*)

Delete a child of an *inx* object with a specified key.

Parameters:

inx a pointer to the *inx* object to be searched.

key the target key of the object to be deleted.

Returns:

true if the delete operation succeeded or false if it did not.

Definition at line 218 of file *inx.c*.

References *inx_auto_unlock()*, *inx_auto_write()*, and *log_pedantic*.

Referenced by *contact_edit()*, *contact_folder_remove()*, *contact_move()*, *imap_folder_remove()*, *imap_message_expunge()*, *message_folder_remove()*, *obj_cache_prune()*, *pop_session_destroy()*, *portal_endpoint_attachments_remove()*, *portal_endpoint_contacts_remove()*, *portal_endpoint_messages_remove()*, *portal_endpoint_messages_send()*, *sess_get()*, and *user_config_edit()*.

5.42.3.18 void* *inx_find* (*inx_t* * *inx*, multi_t *key*)

Find the value associated with a particular key within the children of an *inx* object.

Parameters:

inx a pointer to the *inx* object to be searched.

key the target key to be found.

Returns:

NULL on failure, or the value associated with the requested key on success.

Definition at line 242 of file `inx.c`.

References `inx_auto_read()`, `inx_auto_unlock()`, and `log_pedantic`.

Referenced by `contact_find_detail()`, `contact_find_number()`, `contact_folder_remove()`, `contact_folder_rename()`, `domain_dkim()`, `domain_mailboxes()`, `domain_restricted()`, `domain_spf()`, `domain_wildcard()`, `http_get_static()`, `http_get_template()`, `imap_search_messages()`, `magma_folder_find_number()`, `message_folder_remove()`, `meta_folders_stats_tags()`, `meta_inx_find()`, `meta_inx_remove()`, `meta_messages_mover()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_contacts_load()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_get_upload_attachment()`, `sess_get()`, and `user_config_edit()`.

5.42.3.19 void inx_free (inx_t * inx)

Definition at line 260 of file `inx.c`.

References `inx_auto_unlock()`, `inx_auto_write()`, `log_pedantic`, `mm_free()`, and `rwlock_destroy()`.

Referenced by `contacts_update()`, `http_content_refresh()`, `imap_copy()`, `imap_duplicate_messages()`, `imap_fetch()`, `imap_list()`, `imap_lsub()`, `imap_narrow_messages()`, `imap_search()`, `imap_store()`, `inx_cleanup()`, `inx_cursor_free()`, `magma_folder_fetch()`, `messages_update()`, `meta_data_fetch_alerts()`, `meta_data_fetch_all_tags()`, `meta_update_contacts()`, `meta_update_message_folders()`, `nvp_free()`, `obj_cache_stop()`, `portal_endpoint_alert_list()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_parse_json_str_array()`, `register_blocklist_free()`, `register_blocklist_update()`, `register_data_fetch_blocklist()`, `signature_tree_free()`, `warehouse_fetch_domains()`, and `warehouse_fetch_patterns()`.

5.42.3.20 bool_t inx_insert (inx_t * inx, multi_t key, void * data)

Insert a new record into an `inx` holder.

Parameters:

inx a pointer to the `inx` object that will hold the record.

key a multi-type value specifying the identifier of the record to be inserted.

data a pointer to the data associated with the new key.

Returns:

true if the new record was inserted successfully or false on failure.

Definition at line 165 of file `inx.c`.

References `inx_auto_unlock()`, `inx_auto_write()`, and `log_pedantic`.

Referenced by `contact_details_fetch()`, `contact_folder_create()`, `contact_move()`, `contacts_fetch()`, `http_content_load_fonts()`, `http_data_value_parse()`, `http_load_file()`, `http_parse_header()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_narrow_folders()`, `magma_folder_fetch()`, `message_folder_create()`, `meta_data_fetch_alerts()`, `meta_data_fetch_all_tags()`, `meta_data_fetch_folders()`, `meta_data_fetch_mailbox_aliases()`, `meta_folders_stats_tags()`, `meta_inx_find()`, `meta_messages_copier()`, `nvp_parse()`, `portal_endpoint_attachments_add()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_tag()`, `portal_parse_json_str_array()`, `register_data_fetch_blocklist()`, `sess_create()`, `signature_tree_add()`, `smtp_add_bypass_entry()`, `smtp_fetch_inbound()`, `teacher_add_cookie()`, `user_config_fetch()`, `warehouse_fetch_domains()`, and `warehouse_fetch_patterns()`.

5.42.3.21 void inx_lock_read (inx_t * inx)

Acquire a reader's lock for an `inx` object.

Parameters:

inx a pointer to the inx object to be locked.

Returns:

This function returns no value.

Definition at line 34 of file inx.c.

References `rwlock_lock_read()`.

Referenced by `meta_inx_remove()`, `obj_cache_prune()`, and `sess_get()`.

5.42.3.22 void inx_lock_write (inx_t * inx)

Acquire a writer's lock for an inx object.

Parameters:

inx a pointer to the inx object to be locked.

Returns:

This function returns no value.

Definition at line 53 of file inx.c.

References `rwlock_lock_write()`.

Referenced by `meta_inx_find()`, and `obj_cache_prune()`.

5.42.3.23 uint64_t inx_options (inx_t * inx)

Return the options value of an inx object.

Parameters:

inx a pointer to the inx object to be examined.

Returns:

0 on failure, or the options value of the inx object on success.

Definition at line 72 of file inx.c.

References `inx_auto_read()`, `inx_auto_unlock()`, and `log_pedantic`.

5.42.3.24 bool_t inx_replace (inx_t * inx, multi_t key, void * data)

Replace the value of a key in an inx holder.

Parameters:

inx a pointer to the inx object where the specified key will be replaced.

key a multi-type value specifying the identifier of the record to be replaced.

data a pointer to the new data to be associated with the specified key.

Returns:

true if the specified key's value was replaced successfully, or false on failure.

Definition at line 190 of file `inx.c`.

References `inx_auto_unlock()`, `inx_auto_write()`, and `log_pedantic`.

Referenced by `contact_edit()`, and `user_config_edit()`.

5.42.3.25 `uint64_t inx_serial (inx_t * inx)`

Definition at line 113 of file `inx.c`.

References `inx_auto_read()`, `inx_auto_unlock()`, and `log_pedantic`.

5.42.3.26 `void inx_truncate (inx_t * inx)`

Definition at line 285 of file `inx.c`.

References `inx_auto_unlock()`, `inx_auto_write()`, and `log_pedantic`.

Referenced by `meta_data_fetch_messages()`, and `user_config_update()`.

5.42.3.27 `void inx_unlock (inx_t * inx)`

Unlock an `inx` object.

Parameters:

inx a pointer to the `inx` object to be unlocked.

Returns:

This function returns no value.

Definition at line 15 of file `inx.c`.

References `rwlock_unlock()`.

Referenced by `meta_inx_find()`, `meta_inx_remove()`, `obj_cache_prune()`, and `sess_get()`.

5.42.3.28 `inx_t* linked_alloc (uint64_t options, void * data_free)`

[linked.c](#) [linked.c](#)

Parameters:

options an options value for the newly created linked list object.

data_free a pointer to a function used to free linked list items.

Returns:

NULL on failure or a pointer to the newly allocated linked list object on success.

Definition at line 531 of file `linked.c`.

References `inx_t`, `linked_append()`, `linked_cursor_alloc()`, `linked_cursor_free()`, `linked_cursor_key_active()`, `linked_cursor_key_next()`, `linked_cursor_reset()`, `linked_cursor_value_active()`, `linked_cursor_value_next()`, `linked_delete()`, `linked_find()`, `linked_free()`, `linked_insert()`, `linked_truncate()`, and `mm_alloc()`.

Referenced by `inx_alloc()`.

5.42.4 Variable Documentation

5.42.4.1 inx_cursor_t

Definition at line 66 of file indexes.h.

Referenced by config_load_cmdline_settings(), config_load_file_settings(), contact_find_detail(), contact_find_name(), contacts_fetch(), contacts_update(), http_data_get(), imap_append(), imap_close(), imap_copy(), imap_duplicate_messages(), imap_examine(), imap_expunge(), imap_fetch(), imap_folder_remove(), imap_folder_status(), imap_list(), imap_lsub(), imap_narrow_folders(), imap_narrow_messages(), imap_next_folder_order(), imap_search(), imap_search_messages(), imap_select(), imap_session_destroy(), imap_session_update(), imap_store(), imap_update_flags(), inx_cursor_alloc(), magma_folder_children(), magma_folder_find_full_name(), magma_folder_find_name(), messages_update(), meta_data_fetch_messages(), meta_data_flags_add(), meta_data_flags_remove(), meta_data_flags_replace(), meta_folders_by_name(), meta_folders_by_number(), meta_folders_children(), meta_folders_stats_tags(), meta_message_by_number(), meta_messages_update_sequences(), obj_cache_prune(), pattern_check(), pop_get_last(), pop_get_message(), pop_list(), pop_session_destroy(), pop_session_reset(), pop_total_messages(), pop_total_size(), pop_uidl(), portal_config_collection(), portal_contact_details(), portal_endpoint_alert_list(), portal_endpoint_aliases(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_list(), portal_endpoint_folders_list(), portal_endpoint_folders_tags(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_tag(), portal_endpoint_messages_tags(), portal_smtp_create_data(), portal_smtp_merge_headers(), portal_smtp_relay_message(), portal_upload(), register_abuse_check_blocklist(), signature_tree_get(), signature_tree_verify(), smtp_bypass_check(), smtp_check_filters(), and teacher_print_message().

5.42.4.2 inx_t

Definition at line 62 of file indexes.h.

Referenced by __attribute__(), contact_move(), contacts_update(), hashed_alloc(), hashed_delete(), hashed_find(), hashed_free(), hashed_insert(), hashed_truncate(), imap_copy(), imap_duplicate_messages(), imap_fetch(), imap_list(), imap_lsub(), imap_narrow_folders(), imap_narrow_messages(), imap_search(), imap_search_messages(), imap_store(), inx_alloc(), inx_cursor_free(), linked_alloc(), linked_append(), linked_delete(), linked_find(), linked_free(), linked_insert(), linked_truncate(), magma_folder_fetch(), messages_update(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_folders_stats_tags(), meta_update_contacts(), meta_update_message_folders(), pattern_update(), portal_endpoint_alert_list(), portal_endpoint_folders_tags(), portal_endpoint_messages_flag(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_endpoint_messages_tags(), portal_parse_json_str_array(), register_blocklist_update(), register_data_fetch_blocklist(), tree_alloc(), tree_count(), tree_delete(), tree_find(), tree_free(), tree_insert(), tree_truncate(), warehouse_fetch_domains(), and warehouse_fetch_patterns().

5.43 src/core/indexes/inx.c File Reference

```
#include "magma.h"
```

Functions

- void [inx_unlock](#) ([inx_t](#) *inx)
Unlock an inx object.
- void [inx_auto_unlock](#) ([inx_t](#) *inx)
- void [inx_lock_read](#) ([inx_t](#) *inx)
Acquire a reader's lock for an inx object.
- void [inx_auto_read](#) ([inx_t](#) *inx)
- void [inx_lock_write](#) ([inx_t](#) *inx)
Acquire a writer's lock for an inx object.
- void [inx_auto_write](#) ([inx_t](#) *inx)
- [uint64_t](#) [inx_options](#) ([inx_t](#) *inx)
Return the options value of an inx object.
- [uint64_t](#) [inx_count](#) ([inx_t](#) *inx)
Return the total number of items held by an inx object.
- [uint64_t](#) [inx_serial](#) ([inx_t](#) *inx)
- [bool_t](#) [inx_append](#) ([inx_t](#) *inx, [multi_t](#) key, void *data)
Append a new record onto an inx holder.
- [bool_t](#) [inx_insert](#) ([inx_t](#) *inx, [multi_t](#) key, void *data)
Insert a new record into an inx holder.
- [bool_t](#) [inx_replace](#) ([inx_t](#) *inx, [multi_t](#) key, void *data)
Replace the value of a key in an inx holder.
- [bool_t](#) [inx_delete](#) ([inx_t](#) *inx, [multi_t](#) key)
Delete a child of an inx object with a specified key.
- void * [inx_find](#) ([inx_t](#) *inx, [multi_t](#) key)
Find the value associated with a particular key within the children of an inx object.
- void [inx_free](#) ([inx_t](#) *inx)
- void [inx_truncate](#) ([inx_t](#) *inx)
- void [inx_cleanup](#) ([inx_t](#) *inx)
Perform a checked free of an inx object.
- [inx_t](#) * [inx_alloc](#) ([uint64_t](#) options, void *data_free)
Allocate a new inx instance.

5.43.1 Function Documentation

5.43.1.1 `inx_t* inx_alloc (uint64_t options, void * data_free)`

Allocate a new inx instance. [inx.c](#)

Parameters:

options a value indicating the inx type. Can be M_INX_TREE for a binary tree, M_INX_LINKED for a linked list, or M_INX_HASHED for a hash tree.

data_free a function pointer to a routine to free the data associated with an inx record.

Returns:

NULL on failure or a pointer to the newly created inx object on success.

Definition at line 319 of file inx.c.

References `hashed_alloc()`, `inx_t`, `linked_alloc()`, `log_options`, `M_INX_HASHED`, `M_INX_LINKED`, `M_INX_LOCK_MANUAL`, `M_INX_TREE`, `M_LOG_ERROR`, `M_LOG_STACK_TRACE`, `MAGMA_INDEX_TYPE`, `rwlock_init()`, and `tree_alloc()`.

Referenced by `contact_alloc()`, `contact_folder_alloc()`, `http_content_refresh()`, `http_content_start()`, `http_parse_header()`, `http_parse_pairs()`, `imap_duplicate_messages()`, `imap_narrow_folders()`, `imap_narrow_messages()`, `imap_search_messages()`, `magma_folder_fetch()`, `message_folder_alloc()`, `meta_data_fetch_alerts()`, `meta_data_fetch_all_tags()`, `meta_data_fetch_folders()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_messages()`, `meta_folders_stats_tags()`, `nvp_alloc()`, `obj_cache_start()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_tag()`, `portal_parse_json_str_array()`, `register_data_fetch_blocklist()`, `sess_create()`, `signature_tree_alloc()`, `smtp_add_bypass_entry()`, `smtp_fetch_inbound()`, `teacher_add_cookie()`, `user_config_alloc()`, `warehouse_fetch_domains()`, and `warehouse_fetch_patterns()`.

5.43.1.2 `bool_t inx_append (inx_t * inx, multi_t key, void * data)`

Append a new record onto an inx holder.

Note:

This function only provides benefits for some inx types (like linked lists). For other index types, it simply functions as an alias for the insert function.

Parameters:

inx a pointer to the inx object that will hold the record.

key a multi-type value specifying the identifier of the record to be inserted.

data a pointer to the data associated with the new key.

Returns:

true if the new record was inserted successfully or false on failure.

Definition at line 140 of file inx.c.

References `inx_auto_unlock()`, `inx_auto_write()`, and `log_pedantic`.

Referenced by `imap_append_message()`, `imap_duplicate_messages()`, `imap_message_copier()`, `imap_narrow_messages()`, `imap_search_messages()`, `meta_data_fetch_folder_messages()`, and `meta_data_fetch_messages()`.

5.43.1.3 `void inx_auto_read (inx_t * inx)`

Definition at line 41 of file inx.c.

References `rwlock_lock_read()`.

Referenced by `inx_count()`, `inx_cursor_key_active()`, `inx_cursor_key_next()`, `inx_cursor_value_active()`, `inx_cursor_value_next()`, `inx_find()`, `inx_options()`, and `inx_serial()`.

5.43.1.4 void `inx_auto_unlock` (`inx_t *inx`)

Definition at line 22 of file `inx.c`.

References `rwlock_unlock()`.

Referenced by `inx_append()`, `inx_count()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_key_active()`, `inx_cursor_key_next()`, `inx_cursor_value_active()`, `inx_cursor_value_next()`, `inx_delete()`, `inx_find()`, `inx_free()`, `inx_insert()`, `inx_options()`, `inx_replace()`, `inx_serial()`, and `inx_truncate()`.

5.43.1.5 void `inx_auto_write` (`inx_t *inx`)

Definition at line 60 of file `inx.c`.

References `rwlock_lock_write()`.

Referenced by `inx_append()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_delete()`, `inx_free()`, `inx_insert()`, `inx_replace()`, and `inx_truncate()`.

5.43.1.6 void `inx_cleanup` (`inx_t *inx`)

Perform a checked free of an `inx` object.

See also:

[`inx_free\(\)`](#)

Parameters:

inx the `inx` instance to be freed.

Definition at line 306 of file `inx.c`.

References `inx_free()`.

Referenced by `contact_free()`, `contacts_update()`, `domain_stop()`, `http_content_stop()`, `http_session_reset()`, `imap_narrow_messages()`, `magma_folder_free()`, `messages_update()`, `meta_data_fetch_folders()`, `meta_data_fetch_mailbox_aliases()`, `meta_free()`, `pattern_stop()`, `pattern_update()`, `sess_destroy()`, `sess_release_composition()`, `smtp_free_inbound()`, and `user_config_free()`.

5.43.1.7 uint64_t `inx_count` (`inx_t *inx`)

Return the total number of items held by an `inx` object.

Parameters:

inx a pointer to the `inx` object to be examined.

Returns:

the total number of items held by the `inx` object.

Definition at line 95 of file `inx.c`.

References `count`, `inx_auto_read()`, `inx_auto_unlock()`, and `log_pedantic`.

Referenced by `contact_folder_remove()`, `imap_copy()`, `imap_login()`, `imap_narrow_messages()`, `message_folder_remove()`, `obj_cache_prune()`, `pop_pass()`, `signature_tree_add()`, `signature_tree_get()`, and `signature_tree_verify()`.

5.43.1.8 bool_t `inx_delete` (`inx_t *inx`, `multi_t key`)

Delete a child of an `inx` object with a specified key.

Parameters:

inx a pointer to the inx object to be searched.

key the target key of the object to be deleted.

Returns:

true if the delete operation succeeded or false if it did not.

Definition at line 218 of file inx.c.

References inx_auto_unlock(), inx_auto_write(), and log_pedantic.

Referenced by contact_edit(), contact_folder_remove(), contact_move(), imap_folder_remove(), imap_message_expunge(), message_folder_remove(), obj_cache_prune(), pop_session_destroy(), portal_endpoint_attachments_remove(), portal_endpoint_contacts_remove(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), sess_get(), and user_config_edit().

5.43.1.9 void* inx_find (inx_t * inx, multi_t key)

Find the value associated with a particular key within the children of an inx object.

Parameters:

inx a pointer to the inx object to be searched.

key the target key to be found.

Returns:

NULL on failure, or the value associated with the requested key on success.

Definition at line 242 of file inx.c.

References inx_auto_read(), inx_auto_unlock(), and log_pedantic.

Referenced by contact_find_detail(), contact_find_number(), contact_folder_remove(), contact_folder_rename(), domain_dkim(), domain_mailboxes(), domain_restricted(), domain_spf(), domain_wildcard(), http_get_static(), http_get_template(), imap_search_messages(), magma_folder_find_number(), message_folder_remove(), meta_folders_stats_tags(), meta_inx_find(), meta_inx_remove(), meta_messages_mover(), portal_endpoint_attachments_add(), portal_endpoint_attachments_remove(), portal_endpoint_contacts_load(), portal_endpoint_messages_compose(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_load(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_get_upload_attachment(), sess_get(), and user_config_edit().

5.43.1.10 void inx_free (inx_t * inx)

Definition at line 260 of file inx.c.

References inx_auto_unlock(), inx_auto_write(), log_pedantic, mm_free(), and rwlock_destroy().

Referenced by contacts_update(), http_content_refresh(), imap_copy(), imap_duplicate_messages(), imap_fetch(), imap_list(), imap_lsub(), imap_narrow_messages(), imap_search(), imap_store(), inx_cleanup(), inx_cursor_free(), magma_folder_fetch(), messages_update(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_update_contacts(), meta_update_message_folders(), nvp_free(), obj_cache_stop(), portal_endpoint_alert_list(), portal_endpoint_folders_tags(), portal_endpoint_messages_flag(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_endpoint_messages_tags(), portal_parse_json_str_array(), register_blocklist_free(), register_blocklist_update(), register_data_fetch_blocklist(), signature_tree_free(), warehouse_fetch_domains(), and warehouse_fetch_patterns().

5.43.1.11 bool_t inx_insert (inx_t * inx, multi_t key, void * data)

Insert a new record into an inx holder.

Parameters:

- inx* a pointer to the inx object that will hold the record.
- key* a multi-type value specifying the identifier of the record to be inserted.
- data* a pointer to the data associated with the new key.

Returns:

- true if the new record was inserted successfully or false on failure.

Definition at line 165 of file inx.c.

References inx_auto_unlock(), inx_auto_write(), and log_pedantic.

Referenced by contact_details_fetch(), contact_folder_create(), contact_move(), contacts_fetch(), http_content_load_fonts(), http_data_value_parse(), http_load_file(), http_parse_header(), imap_folder_create(), imap_folder_rename(), imap_narrow_folders(), magma_folder_fetch(), message_folder_create(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_data_fetch_folders(), meta_data_fetch_mailbox_aliases(), meta_folders_stats_tags(), meta_inx_find(), meta_messages_copier(), nvp_parse(), portal_endpoint_attachments_add(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_messages_compose(), portal_endpoint_messages_flag(), portal_endpoint_messages_tag(), portal_parse_json_str_array(), register_data_fetch_blocklist(), sess_create(), signature_tree_add(), smtp_add_bypass_entry(), smtp_fetch_inbound(), teacher_add_cookie(), user_config_fetch(), warehouse_fetch_domains(), and warehouse_fetch_patterns().

5.43.1.12 void inx_lock_read (inx_t * inx)

Acquire a reader's lock for an inx object.

Parameters:

- inx* a pointer to the inx object to be locked.

Returns:

- This function returns no value.

Definition at line 34 of file inx.c.

References rwlock_lock_read().

Referenced by meta_inx_remove(), obj_cache_prune(), and sess_get().

5.43.1.13 void inx_lock_write (inx_t * inx)

Acquire a writer's lock for an inx object.

Parameters:

- inx* a pointer to the inx object to be locked.

Returns:

- This function returns no value.

Definition at line 53 of file inx.c.

References rwlock_lock_write().

Referenced by meta_inx_find(), and obj_cache_prune().

5.43.1.14 uint64_t `inx_options` (`inx_t *inx`)

Return the options value of an `inx` object.

Parameters:

inx a pointer to the `inx` object to be examined.

Returns:

0 on failure, or the options value of the `inx` object on success.

Definition at line 72 of file `inx.c`.

References `inx_auto_read()`, `inx_auto_unlock()`, and `log_pedantic`.

5.43.1.15 bool_t `inx_replace` (`inx_t *inx`, `multi_t key`, `void *data`)

Replace the value of a key in an `inx` holder.

Parameters:

inx a pointer to the `inx` object where the specified key will be replaced.

key a multi-type value specifying the identifier of the record to be replaced.

data a pointer to the new data to be associated with the specified key.

Returns:

true if the specified key's value was replaced successfully, or false on failure.

Definition at line 190 of file `inx.c`.

References `inx_auto_unlock()`, `inx_auto_write()`, and `log_pedantic`.

Referenced by `contact_edit()`, and `user_config_edit()`.

5.43.1.16 uint64_t `inx_serial` (`inx_t *inx`)

Definition at line 113 of file `inx.c`.

References `inx_auto_read()`, `inx_auto_unlock()`, and `log_pedantic`.

5.43.1.17 void `inx_truncate` (`inx_t *inx`)

Definition at line 285 of file `inx.c`.

References `inx_auto_unlock()`, `inx_auto_write()`, and `log_pedantic`.

Referenced by `meta_data_fetch_messages()`, and `user_config_update()`.

5.43.1.18 void `inx_unlock` (`inx_t *inx`)

Unlock an `inx` object.

Parameters:

inx a pointer to the `inx` object to be unlocked.

Returns:

This function returns no value.

Definition at line 15 of file `inx.c`.

References `rwlock_unlock()`.

Referenced by `meta_inx_find()`, `meta_inx_remove()`, `obj_cache_prune()`, and `sess_get()`.

5.44 src/core/indexes/linked.c File Reference

```
#include "magma.h"
```

Functions

- struct `__attribute__((packed))`
- void * `linked_record_get_data` (`linked_record_t` *record, size_t element)
Get the data associated with a linked list record.
- multi_t `linked_record_get_key` (`linked_record_t` *record)
Get the multi-type key of a linked list record.
- void `linked_record_free` (`inx_t` *index, `linked_record_t` *record)
Free a linked list record object and its underlying data.
- `linked_record_t` * `linked_record_alloc` (`multi_t` key, void *data)
Create a new linked list record object.
- void * `linked_find` (void *inx, `multi_t` key)
Find a record in a linked list by key.
- `bool_t` `linked_delete` (void *inx, `multi_t` key)
Remove a record from a linked list and free it and its underlying data.
- `bool_t` `linked_insert` (void *inx, `multi_t` key, void *data)
Create and append a new record to the end of a linked list.
- `bool_t` `linked_append` (void *inx, `multi_t` key, void *data)
Create and append a new record to the end of a linked list.
- `linked_node_t` * `linked_cursor_active` (`linked_cursor_t` *cursor)
Get the current node pointed to by a linked list cursor.
- `linked_node_t` * `linked_cursor_next` (`linked_cursor_t` *cursor)
Get the next node of a linked list cursor.
- void * `linked_cursor_value_next` (`linked_cursor_t` *cursor)
Get the data of a linked list cursor's next record, and update the cursor.
- void * `linked_cursor_value_active` (`linked_cursor_t` *cursor)
Get the data of a linked list cursor's current record.
- `multi_t` `linked_cursor_key_next` (`linked_cursor_t` *cursor)
Get the multi-type key value of a linked list cursor's next record, and update the cursor.
- `multi_t` `linked_cursor_key_active` (`linked_cursor_t` *cursor)
Get the multi-type key value at the current linked list cursor position.
- void `linked_cursor_reset` (`linked_cursor_t` *cursor)
Reset the position of a linked list cursor.

- void [linked_cursor_free](#) ([linked_cursor_t](#) *cursor)
Free a linked list cursor.
- void * [linked_cursor_alloc](#) ([inx_t](#) *inx)
Allocate a cursor to traverse a linked list.
- void [linked_truncate](#) (void *inx)
Truncate all the nodes in a linked list, but do not free it.
- void [linked_free](#) (void *inx)
Free a linked list object and all of its records.
- [inx_t](#) * [linked_alloc](#) (uint64_t options, void *data_free)
Allocate a new linked list instance.

Variables

- [linked_record_t](#)
- [linked_node_t](#)
- [linked_cursor_t](#)

5.44.1 Function Documentation

5.44.1.1 struct __attribute__((packed)) [read]

Definition at line 22 of file [linked.c](#).

References [count](#), [inx_t](#), and [linked_node_t](#).

5.44.1.2 inx_t* linked_alloc (uint64_t options, void * data_free)

Allocate a new linked list instance. [linked.c](#)

Parameters:

options an options value for the newly created linked list object.

data_free a pointer to a function used to free linked list items.

Returns:

NULL on failure or a pointer to the newly allocated linked list object on success.

Definition at line 531 of file [linked.c](#).

References [inx_t](#), [linked_append\(\)](#), [linked_cursor_alloc\(\)](#), [linked_cursor_free\(\)](#), [linked_cursor_key_active\(\)](#), [linked_cursor_key_next\(\)](#), [linked_cursor_reset\(\)](#), [linked_cursor_value_active\(\)](#), [linked_cursor_value_next\(\)](#), [linked_delete\(\)](#), [linked_find\(\)](#), [linked_free\(\)](#), [linked_insert\(\)](#), [linked_truncate\(\)](#), and [mm_alloc\(\)](#).

Referenced by [inx_alloc\(\)](#).

5.44.1.3 `bool_t linked_append(void * inx, multi_t key, void * data)`

Create and append a new record to the end of a linked list.

Note:

This function depends on the *inx* layer to track the list ending via the last variable.

Parameters:

inx a pointer to the linked list that will store the new record.

key a multi-type key value that will be associated with the newly created record.

data a pointer to the data that will be associated with the new record.

Returns:

true on success or false on failure.

Definition at line 244 of file linked.c.

References `inx_t`, `linked_node_t`, `linked_record_alloc()`, `log_info`, `log_pedantic`, `mm_alloc()`, and `mm_free()`.

Referenced by `linked_alloc()`.

5.44.1.4 `linked_node_t* linked_cursor_active(linked_cursor_t * cursor)`

Get the current node pointed to by a linked list cursor.

Parameters:

cursor a pointer to the linked list cursor to be queried.

Returns:

a pointer to the current node indicated by the specified linked list cursor.

LOW: The logic used to find our place in an index that has been modified could be improved.

- If the index we sort the index keys, we could look for the next highest key value.
- We could store the previous node (or even the previous 10 nodes), and try searching for them instead.
- We could also add a delete path through the cursor that would include positional updates.
- We could simply make a deep copy of the entire list so that updates to the original index don't affect the iteration, or use the copy to reconcile.

Definition at line 291 of file linked.c.

References `linked_node_t`.

Referenced by `linked_cursor_key_active()`, `linked_cursor_next()`, and `linked_cursor_value_active()`.

5.44.1.5 `void* linked_cursor_alloc(inx_t * inx)`

Allocate a cursor to traverse a linked list.

Parameters:

inx a pointer the linked list object to be traversed by the cursor.

Returns:

NULL on failure, or a cursor pointing to the head of the linked list on success.

Definition at line 461 of file linked.c.

References linked_cursor_t, log_pedantic, and mm_alloc().

Referenced by linked_alloc().

5.44.1.6 void linked_cursor_free (linked_cursor_t * *cursor*)

Free a linked list cursor.

Parameters:

cursor a pointer to the linked list cursor object to be freed.

Returns:

This function returns no value.

Definition at line 447 of file linked.c.

References mm_free().

Referenced by linked_alloc().

5.44.1.7 multi_t linked_cursor_key_active (linked_cursor_t * *cursor*)

Get the multi-type key value at the current linked list cursor position.

Parameters:

cursor a pointer to the linked list cursor to be queried.

Returns:

an empty multi-type key on failure, or the current linked list cursor's record key on success.

Definition at line 416 of file linked.c.

References linked_cursor_active(), linked_node_t, linked_record_get_key(), and mt_get_null().

Referenced by linked_alloc().

5.44.1.8 multi_t linked_cursor_key_next (linked_cursor_t * *cursor*)

Get the multi-type key value of a linked list cursor's next record, and update the cursor.

Parameters:

cursor a pointer to the linked list cursor to be queried.

Returns:

an empty multi-type key on failure, or the linked list cursor's next record key on success.

Definition at line 400 of file linked.c.

References linked_cursor_next(), linked_node_t, linked_record_get_key(), and mt_get_null().

Referenced by linked_alloc().

5.44.1.9 linked_node_t* linked_cursor_next (linked_cursor_t * cursor)

Get the next node of a linked list cursor.

Note:

The cursor position will be reset if the end of the linked list has been reached.

Parameters:

cursor a pointer to the linked list cursor to be queried.

Returns:

a pointer to the next node of the linked list cursor.

Definition at line 348 of file linked.c.

References `linked_cursor_active()`, and `linked_node_t`.

Referenced by `linked_cursor_key_next()`, and `linked_cursor_value_next()`.

5.44.1.10 void linked_cursor_reset (linked_cursor_t * cursor)

Reset the position of a linked list cursor.

Parameters:

cursor a pointer to the linked list cursor object to be reset.

Returns:

This function returns no value.

Definition at line 432 of file linked.c.

Referenced by `linked_alloc()`.

5.44.1.11 void* linked_cursor_value_active (linked_cursor_t * cursor)

Get the data of a linked list cursor's current record.

Parameters:

cursor a pointer to the linked list cursor to be queried.

Returns:

NULL on failure, or a pointer to the data of the linked list cursor's current record.

Definition at line 385 of file linked.c.

References `linked_cursor_active()`, `linked_node_t`, and `linked_record_get_data()`.

Referenced by `linked_alloc()`.

5.44.1.12 void* linked_cursor_value_next (linked_cursor_t * cursor)

Get the data of a linked list cursor's next record, and update the cursor.

Parameters:

cursor a pointer to the linked list cursor to be queried.

Returns:

NULL on failure, or the data of the linked list cursor's next record on success.

Definition at line 370 of file linked.c.

References `linked_cursor_next()`, `linked_node_t`, and `linked_record_get_data()`.

Referenced by `linked_alloc()`.

5.44.1.13 bool_t linked_delete (void * *inx*, multi_t *key*)

Remove a record from a linked list and free it and its underlying data.

Parameters:

inx a pointer to the linked list to be searched for the specified key.

key a multi-type key value to lookup the record that will be deleted from the linked list.

Returns:

true on success or false on failure.

Definition at line 136 of file linked.c.

References `ident_mt_mt()`, `inx_t`, `linked_node_t`, `linked_record_free()`, and `mm_free()`.

Referenced by `linked_alloc()`.

5.44.1.14 void* linked_find (void * *inx*, multi_t *key*)

Find a record in a linked list by key.

Parameters:

inx a pointer to the linked list to be searched.

key a multi-type key value to be searched against the contents of the `inx` object.

Returns:

NULL on failure or if the record cannot be found, or a pointer to the data of the matching record on success.

Definition at line 107 of file linked.c.

References `ident_mt_mt()`, `inx_t`, `linked_node_t`, and `linked_record_get_data()`.

Referenced by `linked_alloc()`.

5.44.1.15 void linked_free (void * *inx*)

Free a linked list object and all of its records.

See also:

[linked_truncate\(\)](#)

Parameters:

inx a pointer to the linked list object to be freed.

Returns:

This function returns no value.

Definition at line 511 of file linked.c.

References `inx_t`, and `linked_truncate()`.

Referenced by `linked_alloc()`.

5.44.1.16 bool_t linked_insert (void * *inx*, multi_t *key*, void * *data*)

Create and append a new record to the end of a linked list.

Note:

note In the future this should be updated so the key value is used to insert the record in the correct spot.

Parameters:

inx a pointer to the linked list that will store the new record.

key a multi-type key value that will be associated with the newly created record.

data a pointer to the data that will be associated with the new record.

Returns:

true on success or false on failure.

TODO: We are simply looking for the end. The append variation handles this use case. Insert should be using the key value to find the appropriate place to insert the record.

TODO: When the logic is altered to do a comparison, and insert nodes in the proper place this assignment will need to become a conditional.

Definition at line 192 of file linked.c.

References `inx_t`, `linked_node_t`, `linked_record_alloc()`, `log_info`, `mm_alloc()`, and `mm_free()`.

Referenced by `linked_alloc()`.

5.44.1.17 linked_record_t* linked_record_alloc (multi_t *key*, void * *data*)

Create a new linked list record object.

Parameters:

key the multi-type key value of the record to be used for data searches.

data a pointer to the data to be associated with the record.

Returns:

a pointer to the newly allocated and initialized linked record object.

Definition at line 88 of file linked.c.

References `linked_record_t`, `mm_alloc()`, and `mt_dupe()`.

Referenced by `linked_append()`, and `linked_insert()`.

5.44.1.18 void linked_record_free (inx_t * *index*, linked_record_t * *record*)

Free a linked list record object and its underlying data.

Parameters:

index a pointer to the linked list containing the specified record.

record a pointer to the linked list record to be freed.

Returns:

This function returns no value.

Definition at line 67 of file linked.c.

References log_pedantic, mm_free(), and mt_free().

Referenced by linked_delete(), and linked_truncate().

5.44.1.19 void* linked_record_get_data (linked_record_t * *record*, size_t *element*)

Get the data associated with a linked list record.

Parameters:

record a pointer to the linked list record to be queried.

element the index of the data element to be retrieved. Must be set to zero.

Returns:

NULL on failure, or a pointer to the specified linked list record's data on success.

Definition at line 34 of file linked.c.

References log_pedantic.

Referenced by linked_cursor_value_active(), linked_cursor_value_next(), and linked_find().

5.44.1.20 multi_t linked_record_get_key (linked_record_t * *record*)

Get the multi-type key of a linked list record.

Parameters:

record a pointer to the linked list record to be queried.

Returns:

an empty multi-type key on failure, or the specified record's multi-type key value on success.

Definition at line 49 of file linked.c.

References log_pedantic, and mt_get_null().

Referenced by linked_cursor_key_active(), and linked_cursor_key_next().

5.44.1.21 void linked_truncate (void * *inx*)

Truncate all the nodes in a linked list, but do not free it.

Parameters:

inx a pointer to the linked list object to have all of its records truncated.

Returns:

This function returns no value.

Definition at line 480 of file linked.c.

References `inx_t`, `linked_node_t`, `linked_record_free()`, and `mm_free()`.

Referenced by `linked_alloc()`, and `linked_free()`.

5.44.2 Variable Documentation

5.44.2.1 `linked_cursor_t`

Definition at line 26 of file linked.c.

Referenced by `linked_cursor_alloc()`.

5.44.2.2 `linked_node_t`

Definition at line 20 of file linked.c.

Referenced by `__attribute__()`, `linked_append()`, `linked_cursor_active()`, `linked_cursor_key_active()`, `linked_cursor_key_next()`, `linked_cursor_next()`, `linked_cursor_value_active()`, `linked_cursor_value_next()`, `linked_delete()`, `linked_find()`, `linked_insert()`, and `linked_truncate()`.

5.44.2.3 `linked_record_t`

Definition at line 15 of file linked.c.

Referenced by `linked_record_alloc()`.

5.45 src/core/memory/align.c File Reference

```
#include "magma.h"
```

Functions

- `size_t align` (`size_t alignment`, `size_t length`)
align.c

5.45.1 Function Documentation

5.45.1.1 `size_t align` (`size_t alignment`, `size_t length`)

[align.c](#)

Definition at line 16 of file align.c.

Referenced by `alert_alloc()`, `alias_alloc()`, `contact_alloc()`, `contact_detail_alloc()`, `domain_alloc()`, `magma_folder_alloc()`, `message_alloc()`, `meta_folder_stats_tag_alloc()`, `mm_sec_alloc()`, `st_alloc_opts()`, `st_realloc()`, `user_config_alloc()`, and `user_config_entry_alloc()`.

5.46 src/core/memory/bitwise.c File Reference

```
#include "magma.h"
```

Functions

- [uint_t bitwise_count](#) (uint64_t value)
Count the number of bits that are set in a 64-bit number.
- [uchar_t bitwise_or](#) (uchar_t a, uchar_t b)
Performs a bitwise OR operation on two octets and returns the result as a single octet.
- [uchar_t bitwise_xor](#) (uchar_t a, uchar_t b)
Performs a bitwise XOR operation on two octets and returns the result as a single octet.
- [uchar_t bitwise_and](#) (uchar_t a, uchar_t b)
Performs a bitwise AND operation on two octets and returns the result as a single octet.

5.46.1 Function Documentation

5.46.1.1 uchar_t bitwise_and (uchar_t a, uchar_t b) [inline]

Performs a bitwise AND operation on two octets and returns the result as a single octet.

Definition at line 44 of file bitwise.c.

Referenced by st_and().

5.46.1.2 uint_t bitwise_count (uint64_t value)

Count the number of bits that are set in a 64-bit number. bitwise.c

Parameters:

value an unsigned 64-bit value to be checked.

Returns:

the number of bits in value that are set.

Definition at line 15 of file bitwise.c.

Referenced by hashed_bucket(), slots_count(), and st_valid_opts().

5.46.1.3 uchar_t bitwise_or (uchar_t a, uchar_t b) [inline]

Performs a bitwise OR operation on two octets and returns the result as a single octet.

Definition at line 30 of file bitwise.c.

Referenced by st_or().

5.46.1.4 `uchar_t bitwise_xor(uchar_t a, uchar_t b)` `[inline]`

Performs a bitwise XOR operation on two octets and returns the result as a single octet.

Definition at line 37 of file bitwise.c.

Referenced by `st_xor()`.

5.47 src/core/parsers/bitwise.c File Reference

```
#include "magma.h"
```

Functions

- `stringer_t * st_bitwise (stringer_t *a, stringer_t *b, stringer_t *output, uchr_t(*bitwise)(uchr_t, uchr_t))`
Perform bitwise operation on two input strings.
- `stringer_t * st_not (stringer_t *s, stringer_t *output)`
Perform bitwise NOT operation on a managed string.
- `stringer_t * st_or (stringer_t *a, stringer_t *b, stringer_t *output)`
Perform bitwise OR operation between two input strings.
- `stringer_t * st_xor (stringer_t *a, stringer_t *b, stringer_t *output)`
Perform bitwise XOR operation between two input strings.
- `stringer_t * st_and (stringer_t *a, stringer_t *b, stringer_t *output)`
Perform bitwise AND operation between two input strings.

5.47.1 Function Documentation

5.47.1.1 `stringer_t* st_and (stringer_t * a, stringer_t * b, stringer_t * output)`

Perform bitwise AND operation between two input strings.

Parameters:

- a*** First stringer input.
- b*** Second stringer input.
- output*** Stringer in which the result is stored, if no stringer is provided (`output = NULL`) then a new stringer will be allocated for the result.

Returns:

Pointer to output if a valid output stringer was provided, Pointer to a newly allocated stringer if no stringer was specified for the output, NULL on error.

Definition at line 149 of file bitwise.c.

References `bitwise_and()`, and `st_bitwise()`.

5.47.1.2 `stringer_t* st_bitwise (stringer_t * a, stringer_t * b, stringer_t * output, uchr_t(*) (uchr_t, uchr_t) bitwise)`

Perform bitwise operation on two input strings.

Note:

Function pointers are used to determine which bitwise operation is performed. This isn't very efficient, because even though the functions are marked as inline, they aren't actually being inlined by the compiler... but keeping it this way does make it easier to read. If profiling reveals a performance problem, the functions can be moved to the header file, and thus inlined, replaced with a macro, or even a switch statement that keys off an enumerated variable.

Parameters:

- a* First stringer input.
- b* Second stringer input.
- output** Stringer in which the result is stored, if no stringer is provided (output = NULL) then a new stringer will be allocated for the result.
- bitwise** A function pointer to the bitwise operation.

Returns:

Pointer to output if a valid output stringer was provided, or a pointer to a newly allocated stringer if no stringer was specified for the output, NULL on error.

Definition at line 24 of file bitwise.c.

References data, log_pedantic, st_alloc(), st_avail_get(), st_data_get(), st_empty, st_free(), st_length_get(), st_length_set(), st_valid_avail(), and st_valid_destination().

Referenced by st_and(), st_or(), and st_xor().

5.47.1.3 stringer_t* st_not (stringer_t * s, stringer_t * output)

Perform bitwise NOT operation on a managed string.

Parameters:

- s* Stringer input.
- output** Stringer in which the result is stored, if no stringer is provided (output = NULL) then a new stringer will be allocated for the result.

Returns:

Pointer to output if a valid output stringer was provided, or a pointer to a newly allocated stringer if no stringer was specified for the output, NULL on error.

Definition at line 75 of file bitwise.c.

References data, log_pedantic, st_alloc(), st_avail_get(), st_data_get(), st_empty, st_free(), st_length_get(), st_length_set(), st_valid_avail(), and st_valid_destination().

5.47.1.4 stringer_t* st_or (stringer_t * a, stringer_t * b, stringer_t * output)

Perform bitwise OR operation between two input strings.

Parameters:

- a* First stringer input.
- b* Second stringer input.
- output** Stringer in which the result is stored, if no stringer is provided (output = NULL) then a new stringer will be allocated for the result.

Returns:

Pointer to output if a valid output stringer was provided, Pointer to a newly allocated stringer if no stringer was specified for the output, NULL on error.

Definition at line 123 of file bitwise.c.

References bitwise_or(), and st_bitwise().

5.47.1.5 `stringer_t* st_xor (stringer_t * a, stringer_t * b, stringer_t * output)`

Perform bitwise XOR operation between two input strings. bitwise.c

Parameters:

a First stringer input.

b Second stringer input.

output Stringer in which the result is stored, if no stringer is provided (`output = NULL`) then a new stringer will be allocated for the result.

Returns:

Pointer to output if a valid output stringer was provided, or a pointer to a newly allocated stringer if no stringer was specified for the output, `NULL` on error.

Definition at line 136 of file bitwise.c.

References `bitwise_xor()`, and `st_bitwise()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `signature_full_get()`, `signature_full_verify()`, `signature_tree_get()`, `signature_tree_verify()`, `slots_key()`, `slots_set()`, `stacie_decrypt()`, `stacie_encrypt()`, and `stacie_realm_key()`.

5.48 src/core/memory/memory.c File Reference

```
#include "magma.h"
```

Functions

- void [mm_cleanup_variadic](#) (ssize_t len,...)
A checked cleanup function which can be used free a variable number memory buffers.
- [bool_t mm_empty](#) (void *block, size_t len)
Determine whether the memory buffer and/or its length encompass an empty block.
- void * [mm_copy](#) (void *dst, const void *src, size_t len)
Copy data from one buffer to another.
- void * [mm_move](#) (void *dst, void *src, size_t len)
Copy the contents of one buffer into another one, allowing for overlapping data.
- void * [mm_set](#) (void *block, uint8_t set, size_t len)
Sets a block of memory to a specified value.
- void * [mm_wipe](#) (void *block, size_t len)
Zero out a block of memory.
- void [mm_free](#) (void *block)
Free a block of memory.
- void * [mm_dupe](#) (void *block, size_t len)
Duplicate a block of memory.
- void * [mm_alloc](#) (size_t len)
Allocate a chunk of memory with the system allocator and zero-wipe it.

5.48.1 Function Documentation

5.48.1.1 void* mm_alloc (size_t len)

Allocate a chunk of memory with the system allocator and zero-wipe it. [memory.c](#)

Note:

Uses the 'malloc' function attribute to indicate any non-NULL return value is not an alias for any other valid pointer. The buffer length must be non-zero.

See also:

<http://gcc.gnu.org/onlinedocs/gcc-4.4.4/gcc/Function-Attributes.html>

Parameters:

len the amount of memory to allocate.

Returns:

a valid pointer to the allocated memory on success, or NULL on error.

Definition at line 182 of file memory.c.

References `log_pedantic`, and `mm_set()`.

Referenced by `_serialize_ec_privkey()`, `_serialize_ec_pubkey()`, `alert_alloc()`, `alias_alloc()`, `auth_alloc()`, `auth_legacy_alloc()`, `auth_stacie_alloc()`, `cache_alloc()`, `client_connect()`, `compress_alloc()`, `compress_lzo()`, `con_init()`, `con_print()`, `contact_alloc()`, `contact_detail_alloc()`, `cryptex_alloc()`, `deprecated_cryptex_alloc()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_key_public_bin()`, `deprecated_scramble_alloc()`, `dkim_memory_alloc()`, `domain_alloc()`, `ecies_decrypt()`, `ecies_key_public_bin()`, `ed25519_alloc()`, `encrypted_chunk_alloc()`, `encrypted_message_alloc()`, `ephemeral_chunk_alloc()`, `hashed_alloc()`, `hashed_bucket_alloc()`, `hashed_cursor_alloc()`, `http_data_header_parse_line()`, `http_data_value_parse()`, `http_load_file()`, `http_page_get()`, `imap_append_message()`, `imap_copy()`, `imap_fetch_response_add()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_parse_dataitems()`, `keys_alloc()`, `linked_alloc()`, `linked_append()`, `linked_cursor_alloc()`, `linked_insert()`, `linked_record_alloc()`, `magma_folder_alloc()`, `mail_cache_set()`, `mail_create_message()`, `mail_message()`, `mail_mime_part()`, `message_alloc()`, `meta_alloc()`, `meta_data_fetch_folders()`, `meta_data_fetch_messages()`, `meta_folder_stats_tag_alloc()`, `mm_dupe()`, `ns_alloc()`, `ns_dupe()`, `ns_import()`, `nvp_alloc()`, `org_key_alloc()`, `org_key_generate()`, `org_signet_alloc()`, `org_signet_generate()`, `pool_alloc()`, `portal_endpoint_attachments_add()`, `portal_endpoint_messages_compose()`, `prime_alloc()`, `prime_object_alloc()`, `process_start()`, `queue_init()`, `register_session_generate()`, `register_session_get()`, `relay_alloc()`, `requeue()`, `res_bind_create()`, `res_row_store()`, `res_table_alloc()`, `scramble_alloc()`, `servers_alloc()`, `sess_create()`, `signature_tree_alloc()`, `slots_alloc()`, `smtp_add_bypass_entry()`, `smtp_add_recipient()`, `smtp_fetch_authorization()`, `smtp_fetch_inbound()`, `ssl_start()`, `st_alloc_opts()`, `st_realloc()`, `stacker_alloc()`, `stacker_push()`, `stmt_start()`, `system_init_impersonation()`, `tank_start()`, `tank_store()`, `tcp_addr_ip()`, `teacher_data_fetch()`, `teacher_data_get()`, `thread_alloc()`, `tls_print()`, `tree_alloc()`, `tree_cursor_alloc()`, `user_config_alloc()`, `user_config_entry_alloc()`, `user_key_alloc()`, `user_key_generate()`, `user_request_generate()`, `user_request_rotation()`, `user_request_sign()`, and `user_signet_alloc()`.

5.48.1.2 void mm_cleanup_variadic (ssize_t len, ...)

A checked cleanup function which can be used free a variable number memory buffers.

See also:

[mm_free](#)

Parameters:

block the block of memory to be freed.

Returns:

This function returns no value.

Definition at line 18 of file memory.c.

References `mm_free()`.

5.48.1.3 void* mm_copy (void *dst, const void *src, size_t len)

Copy data from one buffer to another.

Note:

For overlapping data regions, `bl_move()` must be used.

Warning:

This function performs an aligned copy so data directly after the `src` buffer may end up inside the `dst` buffer.

Parameters:

dst the destination buffer of the copy operation.

src the source buffer of the copy operation.

len the length, in bytes, of the data to be copied.

Returns:

a pointer to the destination buffer.

Definition at line 59 of file memory.c.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), alert_alloc(), alias_alloc(), chunk_buffer_size(), chunk_header_size(), chunk_header_write(), client_connect(), contact_alloc(), contact_detail_alloc(), deprecated_scramble_encrypt(), deserialize_ns(), deserialize_st(), domain_alloc(), ed25519_generate(), ed25519_private_get(), ed25519_private_set(), ed25519_public_get(), ed25519_public_set(), encrypted_chunk_get(), encrypted_chunk_set(), imap_folder_create(), imap_folder_rename(), imap_parse_literal(), ip_copy(), magma_folder_alloc(), message_alloc(), meta_data_fetch_folders(), meta_data_fetch_messages(), meta_folder_stats_tag_alloc(), mm_dupe(), mm_sec_realloc(), naked_message_set(), net_trigger(), ns_append(), ns_dupe(), ns_import(), prime_field_write(), prime_header_read(), prime_header_write(), prime_reader_size(), res_row_store(), scramble_encrypt(), serialize_ns(), serialize_st(), smtp_auth_plain(), st_append_opts(), st_append_out(), st_copy_in(), st_dupe_opts(), st_import_opts(), st_merge_opts(), st_nullify(), st_realloc(), st_write_variadic(), tank_load(), tank_store(), tcp_addr_ip(), tree_cursor_next(), tree_delete(), and user_config_entry_alloc().

5.48.1.4 void* mm_dupe (void * *block*, size_t *len*)

Duplicate a block of memory.

Parameters:

block a pointer to the block of memory to be duplicated.

len the length, in bytes, of the buffer to be duplicated.

Returns:

a freshly allocated buffer containing a copy of the input data.

Definition at line 153 of file memory.c.

References log_pedantic, mm_alloc(), and mm_copy().

Referenced by contact_name(), imap_command_log_safe(), imap_narrow_folders(), and meta_message_dupe().

5.48.1.5 bool_t mm_empty (void * *block*, size_t *len*)

Determine whether the memory buffer and/or its length encompass an empty block.

Parameters:

block a pointer to the block of memory to be assessed.

len the length, in bytes, of the memory block.

Returns:

false if block is NULL or len is 0; true otherwise.

Definition at line 41 of file memory.c.

Referenced by line_pl_bl(), mm_cmp_ci_eq(), mm_cmp_cs_eq(), str_tok_get_bl(), str_tok_get_count_bl(), tok_get_count_bl(), and tok_get_ns().

5.48.1.6 void mm_free (void * *block*)

Free a block of memory.

Note:

block can point to either a secure or insecure memory block.

Parameters:

block a pointer to the block to be freed.

Returns:

This function does not return any value.

Definition at line 129 of file memory.c.

References `log_pedantic`, and `mm_sec_secured()`.

Referenced by `ar_append()`, `ar_free()`, `auth_free()`, `auth_legacy_free()`, `auth_stacie_free()`, `cache_free()`, `cache_get()`, `cache_get_u64()`, `client_close()`, `client_connect()`, `compress_cleanup()`, `compress_free()`, `compress_lzo()`, `con_destroy()`, `con_init()`, `con_print()`, `config_free()`, `contact_alloc()`, `contact_free()`, `contact_name()`, `deprecated_ecies_key_public_bin()`, `deprecated_scramble_cleanup()`, `deprecated_scramble_free()`, `dequeue()`, `dkim_memory_free()`, `ecies_key_public_bin()`, `ed25519_free()`, `encrypted_chunk_free()`, `encrypted_message_free()`, `ephemeral_chunk_free()`, `hashed_alloc()`, `hashed_cursor_free()`, `hashed_delete()`, `hashed_free()`, `hashed_truncate()`, `http_data_free()`, `http_free_content()`, `http_page_free()`, `imap_append_message()`, `imap_command_log_safe()`, `imap_copy()`, `imap_fetch_free_items()`, `imap_fetch_response_add()`, `imap_fetch_response_free()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_message_copier()`, `imap_narrow_folders()`, `inx_free()`, `keks_free()`, `linked_append()`, `linked_cursor_free()`, `linked_delete()`, `linked_insert()`, `linked_record_free()`, `linked_truncate()`, `magma_folder_alloc()`, `magma_folder_free()`, `mail_cache_destroy()`, `mail_cache_reset()`, `mail_cache_set()`, `mail_create_message()`, `mail_destroy()`, `mail_destroy_message()`, `mail_message()`, `mail_mime_free()`, `message_alloc()`, `meta_alloc()`, `meta_data_fetch_alerts()`, `meta_data_fetch_folders()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_messages()`, `meta_folders_stats_tags()`, `meta_free()`, `meta_message_free()`, `meta_messages_copier()`, `mm_cleanup_variadic()`, `net_listen()`, `ns_free()`, `nvp_alloc()`, `nvp_free()`, `org_key_free()`, `org_signet_free()`, `pool_alloc()`, `pool_free()`, `prime_free()`, `prime_key_generate()`, `prime_object_free()`, `process_start()`, `process_stop()`, `protocol_secure()`, `register_session_free()`, `relay_free()`, `res_bind_free()`, `res_table_free()`, `scramble_cleanup()`, `scramble_free()`, `servers_free()`, `sess_create()`, `sess_destroy()`, `sess_release_attachment()`, `sess_release_composition()`, `signature_tree_free()`, `slots_free()`, `smtp_add_bypass_entry()`, `smtp_add_recipient()`, `smtp_fetch_authorization()`, `smtp_fetch_inbound()`, `smtp_free_inbound()`, `smtp_free_outbound()`, `smtp_free_recipients()`, `smtp_list_free_filter()`, `ssl_stop()`, `st_alloc_opts()`, `st_data_set()`, `st_free()`, `st_realloc()`, `stacker_alloc()`, `stacker_free()`, `stacker_pop()`, `stmt_stop()`, `system_init_impersonation()`, `tank_stop()`, `tank_store()`, `teacher_data_free()`, `thread_alloc()`, `tls_print()`, `tree_alloc()`, `tree_cursor_free()`, `user_config_alloc()`, `user_config_entry_free()`, `user_config_free()`, `user_key_free()`, `user_signet_free()`, `warehouse_fetch_domains()`, and `xml_dump_doc()`.

5.48.1.7 void* mm_move (void *dst, void *src, size_t len)

Copy the contents of one buffer into another one, allowing for overlapping data.

See also:

`memmove()`

Parameters:

dst a pointer to the destination buffer to receive the data to be copied.

src a pointer to the source buffer supplying the data to be copied.

len the length, in bytes, of the data buffer to be copied.

Returns:

a pointer to the specified destination buffer.

Definition at line 72 of file memory.c.

Referenced by `aes_artifact_decrypt()`, `ar_append()`, `auth_response()`, `client_read()`, `client_read_line()`, `con_read()`, `con_read_line()`, `imap_parse_literal()`, `st_trim()`, `stacie_decrypt()`, and `stacie_encrypt()`.

5.48.1.8 void* mm_set (void *block, uint8_t set, size_t len)

Sets a block of memory to a specified value.

Note:

Uses the 'optimize (0)' and 'noinline' function attributes to prevent compiler optimization from removing logic it might consider unnecessary.

Parameters:

block the block of memory to be set.

set the byte value to be written to block.

len the number of times to write the value of the byte repeatedly to block.

Returns:

a pointer to the block of memory passed to the function.

See also:

<http://gcc.gnu.org/onlinedocs/gcc-4.4.4/gcc/Function-Attributes.html>

Definition at line 86 of file memory.c.

Referenced by mm_alloc(), mm_sec_free(), mm_sec_stop(), mm_wipe(), ns_wipe(), st_alloc_opts(), and st_set().

5.48.1.9 void* mm_wipe (void * *block*, size_t *len*)

Zero out a block of memory.

Note:

Uses the 'optimize (0)' and 'noinline' function attributes to prevent compiler optimization from removing logic it might consider unnecessary.

Parameters:

block the block of memory to be zeroed.

len the number of zero bytes to write to memory.

See also:

<http://gcc.gnu.org/onlinedocs/gcc-4.4.4/gcc/Function-Attributes.html>

Definition at line 108 of file memory.c.

References log_pedantic, and mm_set().

Referenced by api_endpoint_delete_user(), ar_alloc(), auth_alloc(), auth_data_fetch(), auth_data_update_legacy(), auth_data_update_lock(), auth_legacy_alloc(), auth_stacie_alloc(), config_fetch_host_number(), config_fetch_settings(), contact_delete(), contact_detail_delete(), contact_detail_upsert(), contact_details_fetch(), contact_insert(), contact_update(), contact_update_stamp(), contacts_fetch(), ed25519_alloc(), encrypted_chunk_alloc(), encrypted_message_alloc(), ephemeral_chunk_alloc(), hex_encode_chr(), http_response_header(), http_response_options(), imap_folder_rename(), imap_folder_status(), keks_alloc(), magma_folder_delete(), magma_folder_fetch(), magma_folder_insert(), magma_folder_rename(), mail_db_delete_message(), mail_db_hide_message(), mail_db_insert_duplicate_message(), mail_db_insert_message(), mail_db_update_message_folder(), meta_alloc(), meta_data_acknowledge_alert(), meta_data_delete_folder(), meta_data_delete_tag(), meta_data_fetch_alerts(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_message_tags(), meta_data_fetch_messages(), meta_data_fetch_shard(), meta_data_fetch_user(), meta_data_flags_add(), meta_data_flags_remove(), meta_data_flags_replace(), meta_data_insert_folder(), meta_data_insert_keys(), meta_data_insert_shard(), meta_data_insert_tag(), meta_data_truncate_tags(), meta_data_update_folder_name(), meta_data_update_log(), mm_sec_alloc(), mm_sec_start(), naked_message_get(), net_init(), net_listen(), net_set_timeout(), net_trigger(),

org_key_alloc(), org_signet_alloc(), portal_message_header(), prime_alloc(), prime_object_alloc(), register_captcha_generate(), register_data_check_username(), register_data_insert_user(), register_session_get(), serv_charset_mysql(), serv_schema_mysql(), serv_version_mysql(), signal_segfault(), signal_shutdown(), signal_start(), signal_thread_start(), signature_tree_alloc(), slots_alloc(), smtp_check_authorized_from(), smtp_check_filters(), smtp_check_rbl(), smtp_check_receive_quota(), smtp_check_transmit_quota(), smtp_fetch_authorization(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_fetch_rollmessages(), smtp_insert_spamsig(), smtp_update_receive_stats(), smtp_update_transmission_stats(), st_trim(), st_wipe(), statistics_init(), stats_init(), stmt_start(), system_init_impersonation(), tank_delete_object(), tank_insert_object(), tank_store(), teacher_data_delete(), teacher_data_fetch(), teacher_data_get(), time_print_gmt(), time_print_local(), user_config_delete(), user_config_fetch(), user_config_upsert(), user_key_alloc(), user_signet_alloc(), virus_engine_refresh(), and virus_start().

5.49 src/core/memory/memory.h File Reference

Defines

- #define [MM_SEC_REQUEST_ALIGNMENT](#) 12
- #define [MM_SEC_PAGE_ALIGNMENT_MIN](#) 1024
- #define [MM_SEC_POOL_LENGTH_MIN](#) 4096
- #define [MEMORYBUF\(l\)](#) (void *)&(([chr_t](#) []){ [0 ... 1] = 0 })
- #define [mm_cleanup\(...\)](#) mm_cleanup_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)

Functions

- [size_t align](#) (size_t alignment, size_t len)
align.c
- [uint_t bitwise_count](#) (uint64_t value)
bitwise.c
- [uchr_t bitwise_or](#) ([uchr_t](#) a, [uchr_t](#) b)
Performs a bitwise OR operation on two octets and returns the result as a single octet.
- [uchr_t bitwise_xor](#) ([uchr_t](#) a, [uchr_t](#) b)
Performs a bitwise XOR operation on two octets and returns the result as a single octet.
- [uchr_t bitwise_and](#) ([uchr_t](#) a, [uchr_t](#) b)
Performs a bitwise AND operation on two octets and returns the result as a single octet.
- void [mm_sec_stop](#) (void)
Deallocate and perform a multi-stage secure wipe of the secure memory region.
- [bool_t mm_sec_start](#) (void)
If enabled, allocate and initialize the secure memory slab.
- void [mm_sec_free](#) (void *block)
Free a secure memory block and perform a multi-pass wipe of its contents.
- void [mm_sec_cleanup](#) (void *block)
Performed a checked memory free.
- [bool_t mm_sec_secured](#) (void *block)
Determine whether the data pointer falls within the secure memory block.
- void * [mm_sec_alloc](#) (size_t len)
Allocate a chunk of memory from the secure memory slab.
- void * [mm_sec_realloc](#) (void *orig, size_t len)
Allocates a larger block of secure memory if requested. Depends on allocation/free routines to lock the required mutex. If a new block is allocated, the original data is copied and then the block is freed. In the event of an error, the original block is preserved and NULL is returned.
- [bool_t mm_sec_stats](#) (size_t *total, size_t *bytes, size_t *items)
Get the collected secure memory statistics for the caller.
- void * [mm_alloc](#) (size_t len)

memory.c

- void **mm_cleanup_variadic** (ssize_t len,...)
A checked cleanup function which can be used free a variable number memory buffers.
- void * **mm_copy** (void *dst, const void *src, size_t len)
Copy data from one buffer to another.
- void * **mm_dup** (void *block, size_t len)
Duplicate a block of memory.
- **bool_t mm_empty** (void *block, size_t len)
Determine whether the memory buffer and/or its length encompass an empty block.
- void **mm_free** (void *block)
Free a block of memory.
- void * **mm_move** (void *dst, void *src, size_t len)
Copy the contents of one buffer into another one, allowing for overlapping data.
- void * **mm_set** (void *block, uint8_t set, size_t len)
Sets a block of memory to a specified value.
- void * **mm_wipe** (void *block, size_t len)
Zero out a block of memory.

5.49.1 Define Documentation

5.49.1.1 #define MEMORYBUF(l) (void *)&((chr_t []){ [0 ... l] = 0 })

Definition at line 51 of file memory.h.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), client_connect(), client_read(), client_read_line(), client_write(), con_reverse_lookup(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_ecies_group(), deprecated_ecies_key_alloc(), deprecated_ecies_key_create(), deprecated_ecies_key_private(), deprecated_ecies_key_private_bin(), deprecated_ecies_key_private_hex(), deprecated_ecies_key_public(), deprecated_ecies_key_public_bin(), deprecated_ecies_key_public_hex(), deprecated_symmetric_decrypt(), dh_exchange_2048(), dh_exchange_4096(), dh_params_generate(), digest_id(), digest_length_output(), digest_name(), ecies_decrypt(), ecies_encrypt(), ecies_group(), ecies_key_alloc(), ecies_key_create(), ecies_key_private(), ecies_key_private_bin(), ecies_key_private_hex(), ecies_key_public(), ecies_key_public_bin(), ecies_key_public_hex(), ed25519_sign(), ed25519_verify(), file_read(), folder_count(), hash_digest(), http_content_load_directory(), http_content_load_fonts(), http_load_file(), ip_presentation(), log_rotate(), log_start(), mm_sec_start(), mutex_destroy(), mutex_init(), mutex_lock(), mutex_unlock(), net_trigger(), prime_start(), process_find_pid(), process_kill(), secp256k1_alloc(), secp256k1_compute_kek(), secp256k1_generate(), secp256k1_private_get(), secp256k1_private_set(), secp256k1_public_get(), secp256k1_public_set(), sess_create(), sess_get(), signal_shutdown(), smtp_check_filters(), smtp_rcpt_to(), spool_check_file(), spool_mktemp(), ssl_verify_privkey(), st_alloc_opts(), st_avail_get(), st_avail_set(), st_data_get(), st_data_set(), st_dup_opts(), st_free(), st_import_opts(), st_length_get(), st_length_set(), st_merge_opts(), st_opt_set(), st_opt_test(), st_realloc(), st_set(), st_vaprint_opts(), st_vsprint(), st_wipe(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_realm_key(), symmetric_decrypt(), system_init_resource_limits(), tcp_addr_ip(), tcp_continue(), tcp_error(), tls_client_alloc(), tls_continue(), tls_error(), and tls_server_alloc().

5.49.1.2 #define mm_cleanup(...) mm_cleanup_variadic(va_narg(__VA_ARGS__), ## __VA_ARGS__)

Definition at line 53 of file memory.h.

Referenced by client_close(), client_connect(), con_destroy(), contact_detail_free(), contact_folder_alloc(), contact_print_form(), imap_fetch_free_items(), message_folder_alloc(), message_free(), queue_shutdown(), and res_bind_free().

5.49.1.3 #define MM_SEC_PAGE_ALIGNMENT_MIN 1024

Definition at line 45 of file memory.h.

Referenced by mm_sec_start().

5.49.1.4 #define MM_SEC_POOL_LENGTH_MIN 4096

Definition at line 48 of file memory.h.

Referenced by mm_sec_start().

5.49.1.5 #define MM_SEC_REQUEST_ALIGNMENT 12

Definition at line 42 of file memory.h.

5.49.2 Function Documentation**5.49.2.1 size_t align (size_t alignment, size_t len)**

[align.c](#)

Definition at line 16 of file align.c.

Referenced by alert_alloc(), alias_alloc(), contact_alloc(), contact_detail_alloc(), domain_alloc(), magma_folder_alloc(), message_alloc(), meta_folder_stats_tag_alloc(), mm_sec_alloc(), st_alloc_opts(), st_realloc(), user_config_alloc(), and user_config_entry_alloc().

5.49.2.2 uchr_t bitwise_and (uchr_t a, uchr_t b) [inline]

Performs a bitwise AND operation on two octets and returns the result as a single octet.

Definition at line 44 of file bitwise.c.

Referenced by st_and().

5.49.2.3 uint_t bitwise_count (uint64_t value)

bitwise.c bitwise.c

Parameters:

value an unsigned 64-bit value to be checked.

Returns:

the number of bits in value that are set.

Definition at line 15 of file bitwise.c.

Referenced by hashed_bucket(), slots_count(), and st_valid_opts().

5.49.2.4 uchr_t bitwise_or (uchr_t a, uchr_t b) [inline]

Performs a bitwise OR operation on two octets and returns the result as a single octet.

Definition at line 30 of file bitwise.c.

Referenced by st_or().

5.49.2.5 uchr_t bitwise_xor (uchr_t a, uchr_t b) [inline]

Performs a bitwise XOR operation on two octets and returns the result as a single octet.

Definition at line 37 of file bitwise.c.

Referenced by st_xor().

5.49.2.6 void* mm_alloc (size_t len)

[memory.c](#) [memory.c](#)

Note:

Uses the 'malloc' function attribute to indicate any non-NULL return value is not an alias for any other valid pointer.
The buffer length must be non-zero.

See also:

<http://gcc.gnu.org/onlinedocs/gcc-4.4.4/gcc/Function-Attributes.html>

Parameters:

len the amount of memory to allocate.

Returns:

a valid pointer to the allocated memory on success, or NULL on error.

Definition at line 182 of file memory.c.

References log_pedantic, and mm_set().

Referenced by _serialize_ec_privkey(), _serialize_ec_pubkey(), alert_alloc(), alias_alloc(), auth_alloc(), auth_legacy_alloc(), auth_stacie_alloc(), cache_alloc(), client_connect(), compress_alloc(), compress_lzo(), con_init(), con_print(), contact_alloc(), contact_detail_alloc(), cryptex_alloc(), deprecated_cryptex_alloc(), deprecated_ecies_decrypt(), deprecated_ecies_key_public_bin(), deprecated_scramble_alloc(), dkim_memory_alloc(), domain_alloc(), ecies_decrypt(), ecies_key_public_bin(), ed25519_alloc(), encrypted_chunk_alloc(), encrypted_message_alloc(), ephemeral_chunk_alloc(), hashed_alloc(), hashed_bucket_alloc(), hashed_cursor_alloc(), http_data_header_parse_line(), http_data_value_parse(), http_load_file(), http_page_get(), imap_append_message(), imap_copy(), imap_fetch_response_add(), imap_folder_create(), imap_folder_rename(), imap_parse_dataitems(), keks_alloc(), linked_alloc(), linked_append(), linked_cursor_alloc(), linked_insert(), linked_record_alloc(), magma_folder_alloc(), mail_cache_set(), mail_create_message(), mail_message(), mail_mime_part(), message_alloc(), meta_alloc(), meta_data_fetch_folders(), meta_data_fetch_messages(), meta_folder_stats_tag_alloc(), mm_dupe(), ns_alloc(), ns_dupe(), ns_import(), nvp_alloc(), org_key_alloc(), org_key_generate(), org_signet_alloc(), org_signet_generate(), pool_alloc(), portal_endpoint_attachments_add(), portal_endpoint_messages_compose(), prime_alloc(), prime_object_alloc(), process_start(), queue_init(), register_session_generate(), register_session_get(), relay_alloc(), requeue(), res_bind_create(), res_row_store(), res_table_alloc(), scramble_alloc(), servers_alloc(), sess_create(), signature_tree_alloc(), slots_alloc(), smtp_add_bypass_entry(), smtp_add_recipient(), smtp_fetch_authorization(), smtp_fetch_inbound(), ssl_start(), st_alloc_opts(), st_realloc(), stacker_alloc(), stacker_push(), stmt_start(), system_init_impersonation(), tank_start(), tank_store(), tcp_addr_ip(), teacher_data_fetch(), teacher_data_get(), thread_alloc(), tls_print(), tree_alloc(), tree_cursor_alloc(), user_config_alloc(), user_config_entry_alloc(), user_key_alloc(), user_key_generate(), user_request_generate(), user_request_rotation(), user_request_sign(), and user_signet_alloc().

5.49.2.7 void mm_cleanup_variadic (ssize_t len, ...)

A checked cleanup function which can be used free a variable number memory buffers.

See also:

[mm_free](#)

Parameters:

block the block of memory to be freed.

Returns:

This function returns no value.

Definition at line 18 of file memory.c.

References mm_free().

5.49.2.8 void* mm_copy (void *dst, const void *src, size_t len)

Copy data from one buffer to another.

Note:

For overlapping data regions, bl_move() must be used.

Warning:

This function performs an aligned copy so data directly after the src buffer may end up inside the dst buffer.

Parameters:

dst the destination buffer of the copy operation.

src the source buffer of the copy operation.

len the length, in bytes, of the data to be copied.

Returns:

a pointer to the destination buffer.

Definition at line 59 of file memory.c.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), alert_alloc(), alias_alloc(), chunk_buffer_size(), chunk_header_size(), chunk_header_write(), client_connect(), contact_alloc(), contact_detail_alloc(), deprecated_scramble_encrypt(), deserialize_ns(), deserialize_st(), domain_alloc(), ed25519_generate(), ed25519_private_get(), ed25519_private_set(), ed25519_public_get(), ed25519_public_set(), encrypted_chunk_get(), encrypted_chunk_set(), imap_folder_create(), imap_folder_rename(), imap_parse_literal(), ip_copy(), magma_folder_alloc(), message_alloc(), meta_data_fetch_folders(), meta_data_fetch_messages(), meta_folder_stats_tag_alloc(), mm_dupe(), mm_sec_realloc(), naked_message_set(), net_trigger(), ns_append(), ns_dupe(), ns_import(), prime_field_write(), prime_header_read(), prime_header_write(), prime_reader_size(), res_row_store(), scramble_encrypt(), serialize_ns(), serialize_st(), smtp_auth_plain(), st_append_opts(), st_append_out(), st_copy_in(), st_dupe_opts(), st_import_opts(), st_merge_opts(), st_nullify(), st_realloc(), st_write_variadic(), tank_load(), tank_store(), tcp_addr_ip(), tree_cursor_next(), tree_delete(), and user_config_entry_alloc().

5.49.2.9 void* mm_dupe (void *block, size_t len)

Duplicate a block of memory.

Parameters:

block a pointer to the block of memory to be duplicated.

len the length, in bytes, of the buffer to be duplicated.

Returns:

a freshly allocated buffer containing a copy of the input data.

Definition at line 153 of file memory.c.

References log_pedantic, mm_alloc(), and mm_copy().

Referenced by contact_name(), imap_command_log_safe(), imap_narrow_folders(), and meta_message_dupe().

5.49.2.10 bool_t mm_empty (void * *block*, size_t *len*)

Determine whether the memory buffer and/or its length encompass an empty block.

Parameters:

block a pointer to the block of memory to be assessed.

len the length, in bytes, of the memory block.

Returns:

false if block is NULL or len is 0; true otherwise.

Definition at line 41 of file memory.c.

Referenced by line_pl_bl(), mm_cmp_ci_eq(), mm_cmp_cs_eq(), str_tok_get_bl(), str_tok_get_count_bl(), tok_get_count_bl(), and tok_get_ns().

5.49.2.11 void mm_free (void * *block*)

Free a block of memory.

Note:

block can point to either a secure or insecure memory block.

Parameters:

block a pointer to the block to be freed.

Returns:

This function does not return any value.

Definition at line 129 of file memory.c.

References log_pedantic, and mm_sec_secured().

Referenced by ar_append(), ar_free(), auth_free(), auth_legacy_free(), auth_stacie_free(), cache_free(), cache_get(), cache_get_u64(), client_close(), client_connect(), compress_cleanup(), compress_free(), compress_lzo(), con_destroy(), con_init(), con_print(), config_free(), contact_alloc(), contact_free(), contact_name(), deprecated_ecies_key_public_bin(), deprecated_scramble_cleanup(), deprecated_scramble_free(), dequeue(), dkim_memory_free(), ecies_key_public_bin(), ed25519_free(), encrypted_chunk_free(), encrypted_message_free(), ephemeral_chunk_free(), hashed_alloc(), hashed_cursor_free(), hashed_delete(), hashed_free(), hashed_truncate(), http_data_free(), http_free_content(), http_page_free(), imap_append_message(), imap_command_log_safe(), imap_copy(), imap_fetch_free_items(), imap_fetch_response_add(), imap_fetch_response_free(), imap_folder_create(), imap_folder_rename(), imap_message_copier(), imap_narrow_folders(), inx_free(), keks_free(), linked_append(), linked_cursor_free(), linked_delete(), linked_insert(), linked_record_free(), linked_truncate(), magma_folder_alloc(), magma_folder_free(), mail_cache_destroy(), mail_cache_reset(), mail_cache_set(), mail_create_message(), mail_destroy(), mail_destroy_message(), mail_message(), mail_mime_free(), message_alloc(), meta_alloc(), meta_data_fetch_alerts(), meta_data_fetch_folders(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), meta_folders_stats_tags(), meta_free(), meta_message_free(), meta_messages_copier(), mm_cleanup_variadic(), net_listen(), ns_free(), nvp_alloc(), nvp_free(), org_key_free(), org_signet_free(), pool_alloc(), pool_free(), prime_free(), prime_key_generate(), prime_object_free(), process_start(), process_stop(), protocol_secure(), register_session_free(), relay_free(), res_bind_free(), res_table_free(), scramble_cleanup(), scramble_free(), servers_free(), sess_create(), sess_destroy(), sess_release_attachment(), sess_release_composition(), signature_tree_free(), slots_free(), smtp_add_bypass_entry(), smtp_add_recipient(), smtp_fetch_authorization(), smtp_fetch_inbound(), smtp_free_inbound(), smtp_free_outbound(), smtp_free_recipients(), smtp_list_free_filter(), ssl_stop(), st_alloc_opts(), st_data_set(), st_free(), st_realloc(), stacker_alloc(), stacker_free(), stacker_pop(), stmt_stop(), system_init_impersonation(), tank_stop(), tank_store(), teacher_data_free(), thread_alloc(), tls_print(), tree_alloc(), tree_cursor_free(), user_config_alloc(), user_config_entry_free(), user_config_free(), user_key_free(), user_signet_free(), warehouse_fetch_domains(), and xml_dump_doc().

5.49.2.12 void* mm_move (void * *dst*, void * *src*, size_t *len*)

Copy the contents of one buffer into another one, allowing for overlapping data.

See also:

`memmove()`

Parameters:

dst a pointer to the destination buffer to receive the data to be copied.

src a pointer to the source buffer supplying the data to be copied.

len the length, in bytes, of the data buffer to be copied.

Returns:

a pointer to the specified destination buffer.

Definition at line 72 of file `memory.c`.

Referenced by `aes_artifact_decrypt()`, `ar_append()`, `auth_response()`, `client_read()`, `client_read_line()`, `con_read()`, `con_read_line()`, `imap_parse_literal()`, `st_trim()`, `stacie_decrypt()`, and `stacie_encrypt()`.

5.49.2.13 void* mm_sec_alloc (size_t *len*)

Allocate a chunk of memory from the secure memory slab.

See also:

[mm_sec_chunk_new\(\)](#)

Parameters:

len the length, in bytes, of the secure memory chunk to be allocated.

Returns:

NULL on failure, or a pointer to the freshly allocated chunk of secure memory on success.

Definition at line 251 of file `secure.c`.

References `align()`, `bytes`, `items`, `log_options`, `log_pedantic`, `M_LOG_PEDANTIC`, `M_LOG_STACK_TRACE`, `mm_sec_chunk_new()`, `mm_sec_stats()`, `mm_wipe()`, `mutex_lock()`, `mutex_unlock()`, and `secured_t`.

Referenced by `deprecated_ecies_key_private_bin()`, `ecies_key_private_bin()`, `mm_sec_realloc()`, `ssl_start()`, `st_alloc_opts()`, and `st_realloc()`.

5.49.2.14 void mm_sec_cleanup (void * *block*)

Performed a checked memory free.

See also:

[mm_sec_free](#)

Parameters:

block the block of memory to be freed.

Returns:

This function returns no value.

Definition at line 236 of file `secure.c`.

References `mm_sec_free()`.

5.49.2.15 void mm_sec_free (void * *block*)

Free a secure memory block and perform a multi-pass wipe of its contents.

Returns:

This function returns no value.

Definition at line 196 of file secure.c.

References log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, MM_SEC_CHUNK_ALLOCATED, mm_sec_chunk_merge(), mm_sec_secured(), mm_set(), mutex_lock(), mutex_unlock(), and secured_t.

Referenced by deprecated_ecies_key_private_bin(), ecies_key_private_bin(), mm_sec_cleanup(), mm_sec_realloc(), ssl_start(), st_alloc_opts(), st_data_set(), st_free(), and st_realloc().

5.49.2.16 void* mm_sec_realloc (void * *orig*, size_t *len*)

Allocates a larger block of secure memory if requested. Depends on allocation/free routines to lock the required mutex. If a new block is allocated, the original data is copied and then the block is freed. In the event of an error, the original block is preserved and NULL is returned.

Definition at line 296 of file secure.c.

References log_pedantic, mm_copy(), mm_sec_alloc(), mm_sec_free(), and secured_t.

5.49.2.17 bool_t mm_sec_secured (void * *block*)

Determine whether the data pointer falls within the secure memory block.

Parameters:

block the data pointer to be tested.

Returns:

true if block points to secure data; false if not, or if block is invalid or secure memory is disabled.

Definition at line 80 of file secure.c.

References slab.

Referenced by mm_free(), mm_sec_chunk_new(), mm_sec_chunk_next(), mm_sec_free(), and st_data_set().

5.49.2.18 bool_t mm_sec_start (void)

If enabled, allocate and initialize the secure memory slab.

Note:

This function will mmap a page-aligned secure memory slab (defaults to 32768 bytes long), mlock() it into memory, and zero-wipe it. Guard pages with empty permissions are created on the boundaries of the slab to prevent memory bungling.

Returns:

true if the secure memory slab has been initialized, or false if the process fails.

Definition at line 357 of file secure.c.

References log_pedantic, magma, magma_t::memory, MEMORYBUF, MM_SEC_PAGE_ALIGNMENT_MIN, MM_SEC_POOL_LENGTH_MIN, mm_wipe(), magma_t::page_length, magma_t::secure, and secured_t.

Referenced by process_start().

5.49.2.19 bool_t mm_sec_stats (size_t * total, size_t * bytes, size_t * items)

Get the collected secure memory statistics for the caller.

Parameters:

total a pointer to a size_t variable that will store the secure memory region length, in bytes.

bytes a pointer to a size_t variable that will store the number of secure bytes allocated by magma.

items a pointer to a size_t variable that will store the number of secure memory allocations requested by magma.

Returns:

true on success or false on failure.

Definition at line 60 of file secure.c.

References mutex_lock(), and mutex_unlock().

Referenced by mm_sec_alloc(), and stats_derived_value().

5.49.2.20 void mm_sec_stop (void)

Deallocate and perform a multi-stage secure wipe of the secure memory region.

Returns:

This function returns no value.

Definition at line 328 of file secure.c.

References mm_set().

Referenced by process_stop().

5.49.2.21 void* mm_set (void * block, uint8_t set, size_t len)

Sets a block of memory to a specified value.

Note:

Uses the 'optimize (0)' and 'noinline' function attributes to prevent compiler optimization from removing logic it might consider unnecessary.

Parameters:

block the block of memory to be set.

set the byte value to be written to block.

len the number of times to write the value of the byte repeatedly to block.

Returns:

a pointer to the block of memory passed to the function.

See also:

<http://gcc.gnu.org/onlinedocs/gcc-4.4.4/gcc/Function-Attributes.html>

Definition at line 86 of file memory.c.

Referenced by mm_alloc(), mm_sec_free(), mm_sec_stop(), mm_wipe(), ns_wipe(), st_alloc_opts(), and st_set().

5.49.2.22 void* mm_wipe (void * *block*, size_t *len*)

Zero out a block of memory.

Note:

Uses the 'optimize (0)' and 'noinline' function attributes to prevent compiler optimization from removing logic it might consider unnecessary.

Parameters:

block the block of memory to be zeroed.

len the number of zero bytes to write to memory.

See also:

<http://gcc.gnu.org/onlinedocs/gcc-4.4.4/gcc/Function-Attributes.html>

Definition at line 108 of file memory.c.

References `log_pedantic`, and `mm_set()`.

Referenced by `api_endpoint_delete_user()`, `ar_alloc()`, `auth_alloc()`, `auth_data_fetch()`, `auth_data_update_legacy()`, `auth_data_update_lock()`, `auth_legacy_alloc()`, `auth_stacie_alloc()`, `config_fetch_host_number()`, `config_fetch_settings()`, `contact_delete()`, `contact_detail_delete()`, `contact_detail_upsert()`, `contact_details_fetch()`, `contact_insert()`, `contact_update()`, `contact_update_stamp()`, `contacts_fetch()`, `ed25519_alloc()`, `encrypted_chunk_alloc()`, `encrypted_message_alloc()`, `ephemeral_chunk_alloc()`, `hex_encode_chr()`, `http_response_header()`, `http_response_options()`, `imap_folder_rename()`, `imap_folder_status()`, `keys_alloc()`, `magma_folder_delete()`, `magma_folder_fetch()`, `magma_folder_insert()`, `magma_folder_rename()`, `mail_db_delete_message()`, `mail_db_hide_message()`, `mail_db_insert_duplicate_message()`, `mail_db_insert_message()`, `mail_db_update_message_folder()`, `meta_alloc()`, `meta_data_acknowledge_alert()`, `meta_data_delete_folder()`, `meta_data_delete_tag()`, `meta_data_fetch_alerts()`, `meta_data_fetch_folder_messages()`, `meta_data_fetch_folders()`, `meta_data_fetch_keys()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_message_tags()`, `meta_data_fetch_messages()`, `meta_data_fetch_shard()`, `meta_data_fetch_user()`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_flags_replace()`, `meta_data_insert_folder()`, `meta_data_insert_keys()`, `meta_data_insert_shard()`, `meta_data_insert_tag()`, `meta_data_truncate_tags()`, `meta_data_update_folder_name()`, `meta_data_update_log()`, `mm_sec_alloc()`, `mm_sec_start()`, `naked_message_get()`, `net_init()`, `net_listen()`, `net_set_timeout()`, `net_trigger()`, `org_key_alloc()`, `org_signet_alloc()`, `portal_message_header()`, `prime_alloc()`, `prime_object_alloc()`, `register_captcha_generate()`, `register_data_check_username()`, `register_data_insert_user()`, `register_session_get()`, `serv_charset_mysql()`, `serv_schema_mysql()`, `serv_version_mysql()`, `signal_segfault()`, `signal_shutdown()`, `signal_start()`, `signal_thread_start()`, `signature_tree_alloc()`, `slots_alloc()`, `smtp_check_authorized_from()`, `smtp_check_filters()`, `smtp_check_rbl()`, `smtp_check_receive_quota()`, `smtp_check_transmit_quota()`, `smtp_fetch_authorized()`, `smtp_fetch_autoreply()`, `smtp_fetch_inbound()`, `smtp_fetch_rollmessages()`, `smtp_insert_spamsig()`, `smtp_update_receive_stats()`, `smtp_update_transmission_stats()`, `st_trim()`, `st_wipe()`, `statistics_init()`, `stats_init()`, `stmt_start()`, `system_init_impersonation()`, `tank_delete_object()`, `tank_insert_object()`, `tank_store()`, `teacher_data_delete()`, `teacher_data_fetch()`, `teacher_data_get()`, `time_print_gmt()`, `time_print_local()`, `user_config_delete()`, `user_config_fetch()`, `user_config_upsert()`, `user_key_alloc()`, `user_signet_alloc()`, `virus_engine_refresh()`, and `virus_start()`.

5.50 src/core/memory/secure.c File Reference

```
#include "magma.h"
```

Enumerations

- enum { [MM_SEC_CHUNK_AVAILABLE](#) = 0, [MM_SEC_CHUNK_ALLOCATED](#) = 1 }

Functions

- struct [__attribute__](#) ((packed))
- [bool_t mm_sec_stats](#) (size_t *total, size_t *bytes, size_t *items)
Get the collected secure memory statistics for the caller.
- [bool_t mm_sec_secured](#) (void *block)
Determine whether the data pointer falls within the secure memory block.
- [secured_t * mm_sec_chunk_next](#) (secured_t *chunk)
Get the next chunk of secure memory.
- [secured_t * mm_sec_chunk_prev](#) (secured_t *chunk)
Get the previous chunk of secure memory.
- void [mm_sec_chunk_merge](#) (secured_t *chunk)
If an adjacent region is available, merge them together.
- [secured_t * mm_sec_chunk_new](#) (secured_t *block, size_t size)
Locates a properly sized chunk of memory and reserves it.
- void [mm_sec_free](#) (void *block)
Free a secure memory block and perform a multi-pass wipe of its contents.
- void [mm_sec_cleanup](#) (void *block)
Performed a checked memory free.
- void * [mm_sec_alloc](#) (size_t len)
Allocate a chunk of memory from the secure memory slab.
- void * [mm_sec_realloc](#) (void *orig, size_t len)
Allocates a larger block of secure memory if requested. Depends on allocation/free routines to lock the required mutex. If a new block is allocated, the original data is copied and then the block is freed. In the event of an error, the original block is preserved and NULL is returned.
- void [mm_sec_stop](#) (void)
Deallocate and perform a multi-stage secure wipe of the secure memory region.
- [bool_t mm_sec_start](#) (void)
If enabled, allocate and initialize the secure memory slab.

Variables

- [secured_t](#)

5.50.1 Enumeration Type Documentation

5.50.1.1 anonymous enum

Enumerator:

MM_SEC_CHUNK_AVAILABLE
MM_SEC_CHUNK_ALLOCATED

Definition at line 10 of file secure.c.

5.50.2 Function Documentation

5.50.2.1 struct __attribute__((packed)) [read]

Definition at line 15 of file secure.c.

References length.

5.50.2.2 void* mm_sec_alloc (size_t len)

Allocate a chunk of memory from the secure memory slab.

See also:

[mm_sec_chunk_new\(\)](#)

Parameters:

len the length, in bytes, of the secure memory chunk to be allocated.

Returns:

NULL on failure, or a pointer to the freshly allocated chunk of secure memory on success.

Definition at line 251 of file secure.c.

References align(), bytes, items, log_options, log_pedantic, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, mm_sec_chunk_new(), mm_sec_stats(), mm_wipe(), mutex_lock(), mutex_unlock(), and secured_t.

Referenced by deprecated_ecies_key_private_bin(), ecies_key_private_bin(), mm_sec_realloc(), ssl_start(), st_alloc_opts(), and st_realloc().

5.50.2.3 void mm_sec_chunk_merge (secured_t * chunk)

If an adjacent region is available, merge them together.

Definition at line 135 of file secure.c.

References MM_SEC_CHUNK_ALLOCATED, mm_sec_chunk_next(), mm_sec_chunk_prev(), and secured_t.

Referenced by mm_sec_chunk_new(), and mm_sec_free().

5.50.2.4 secured_t* mm_sec_chunk_new (secured_t * block, size_t size)

Locates a properly sized chunk of memory and reserves it.

Definition at line 157 of file secure.c.

References MM_SEC_CHUNK_ALLOCATED, mm_sec_chunk_merge(), mm_sec_chunk_next(), mm_sec_secured(), and secured_t.

Referenced by mm_sec_alloc().

5.50.2.5 `secured_t* mm_sec_chunk_next (secured_t * chunk)`

Get the next chunk of secure memory.

Parameters:

chunk the input secure chunk.

Returns:

a pointer to the next chunk of secure memory, or NULL on failure or if the end of the slab is reached.

Definition at line 102 of file `secure.c`.

References `mm_sec_secured()`, and `secured_t`.

Referenced by `mm_sec_chunk_merge()`, `mm_sec_chunk_new()`, and `mm_sec_chunk_prev()`.

5.50.2.6 `secured_t* mm_sec_chunk_prev (secured_t * chunk)`

Get the previous chunk of secure memory.

Parameters:

chunk the input secure chunk.

Returns:

a pointer to the previous chunk of secure memory, or NULL on failure or if the beginning of the slab is reached.

Definition at line 117 of file `secure.c`.

References `mm_sec_chunk_next()`, and `secured_t`.

Referenced by `mm_sec_chunk_merge()`.

5.50.2.7 `void mm_sec_cleanup (void * block)`

Performed a checked memory free.

See also:

[mm_sec_free](#)

Parameters:

block the block of memory to be freed.

Returns:

This function returns no value.

Definition at line 236 of file `secure.c`.

References `mm_sec_free()`.

5.50.2.8 `void mm_sec_free (void * block)`

Free a secure memory block and perform a multi-pass wipe of its contents.

Returns:

This function returns no value.

Definition at line 196 of file secure.c.

References log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, MM_SEC_CHUNK_ALLOCATED, mm_sec_chunk_merge(), mm_sec_secured(), mm_set(), mutex_lock(), mutex_unlock(), and secured_t.

Referenced by deprecated_ecies_key_private_bin(), ecies_key_private_bin(), mm_sec_cleanup(), mm_sec_realloc(), ssl_start(), st_alloc_opts(), st_data_set(), st_free(), and st_realloc().

5.50.2.9 void* mm_sec_realloc (void * orig, size_t len)

Allocates a larger block of secure memory if requested. Depends on allocation/free routines to lock the required mutex. If a new block is allocated, the original data is copied and then the block is freed. In the event of an error, the original block is preserved and NULL is returned.

Definition at line 296 of file secure.c.

References log_pedantic, mm_copy(), mm_sec_alloc(), mm_sec_free(), and secured_t.

5.50.2.10 bool_t mm_sec_secured (void * block)

Determine whether the data pointer falls within the secure memory block.

Parameters:

block the data pointer to be tested.

Returns:

true if block points to secure data; false if not, or if block is invalid or secure memory is disabled.

Definition at line 80 of file secure.c.

References slab.

Referenced by mm_free(), mm_sec_chunk_new(), mm_sec_chunk_next(), mm_sec_free(), and st_data_set().

5.50.2.11 bool_t mm_sec_start (void)

If enabled, allocate and initialize the secure memory slab.

Note:

This function will mmap a page-aligned secure memory slab (defaults to 32768 bytes long), mlock() it into memory, and zero-wipe it. Guard pages with empty permissions are created on the boundaries of the slab to prevent memory bungling.

Returns:

true if the secure memory slab has been initialized, or false if the process fails.

Definition at line 357 of file secure.c.

References log_pedantic, magma, magma_t::memory, MEMORYBUF, MM_SEC_PAGE_ALIGNMENT_MIN, MM_SEC_POOL_LENGTH_MIN, mm_wipe(), magma_t::page_length, magma_t::secure, and secured_t.

Referenced by process_start().

5.50.2.12 `bool_t mm_sec_stats (size_t * total, size_t * bytes, size_t * items)`

Get the collected secure memory statistics for the caller.

Parameters:

total a pointer to a `size_t` variable that will store the secure memory region length, in bytes.

bytes a pointer to a `size_t` variable that will store the number of secure bytes allocated by magma.

items a pointer to a `size_t` variable that will store the number of secure memory allocations requested by magma.

Returns:

true on success or false on failure.

Definition at line 60 of file `secure.c`.

References `mutex_lock()`, and `mutex_unlock()`.

Referenced by `mm_sec_alloc()`, and `stats_derived_value()`.

5.50.2.13 `void mm_sec_stop (void)`

Deallocate and perform a multi-stage secure wipe of the secure memory region.

Returns:

This function returns no value.

Definition at line 328 of file `secure.c`.

References `mm_set()`.

Referenced by `process_stop()`.

5.50.3 Variable Documentation

5.50.3.1 `struct { ... } allocated`

5.50.3.2 `size_t bytes`

Definition at line 32 of file `secure.c`.

Referenced by `client_read()`, `client_read_line()`, `client_write()`, `con_print()`, `con_read()`, `con_read_line()`, `con_write_bl()`, `deserialize_ns()`, `mm_sec_alloc()`, `prime_unpack_fields()`, `prime_unpack_validate()`, `stats_derived_value()`, `tls_print()`, `utf8_length_st()`, and `utf8_valid_st()`.

5.50.3.3 `void* data`

Definition at line 24 of file `secure.c`.

Referenced by `__attribute__()`, `_clone_cached_object()`, `_compute_sha_hash_multibuf()`, `bracket_extract_pl()`, `cache_get()`, `cache_get_u64()`, `chunk_buffer_size()`, `chunk_header_size()`, `chunk_header_type()`, `contact_name()`, `contact_print_form()`, `ephemeral_chunk_set()`, `hash_murmur32()`, `hash_murmur64()`, `hashed_bucket_find_key()`, `hashed_find()`, `http_body()`, `http_data_get()`, `http_data_value_parse()`, `http_load_file()`, `http_parse_header()`, `http_parse_pairs()`, `int16_conv_bl()`, `int32_conv_bl()`, `int64_conv_bl()`, `int8_conv_bl()`, `naked_message_get()`, `part_decrypt()`, `part_encrypt()`, `portal_endpoint_messages_load()`, `portal_upload()`, `register_business_step1()`, `register_business_step2()`, `register_session_cache()`, `register_session_get()`, `smtp_check_filters()`, `st_bitwise()`, `st_free()`, `st_length_get()`, `st_not()`, `st_realloc()`, `st_write_variadic()`, `teacher_data_get()`, `teacher_data_save()`, `uint16_conv_bl()`, `uint16_get_no()`, `uint16_put_no()`, `uint24_get_no()`, `uint24_put_no()`, `uint32_conv_bl()`, `uint32_get_no()`, `uint32_put_no()`, `uint64_conv_bl()`, and `uint8_conv_bl()`.

5.50.3.4 void* data_true

Definition at line 23 of file secure.c.

5.50.3.5 bool_t enabled

Definition at line 35 of file secure.c.

5.50.3.6 size_t items

Definition at line 31 of file secure.c.

Referenced by `imap_fetch()`, `imap_fetch_bodystructure()`, `mm_sec_alloc()`, and `stats_derived_value()`.

5.50.3.7 size_t length

Definition at line 25 of file secure.c.

Referenced by `__attribute__()`, `cache_get()`, `cache_get_u64()`, `client_write()`, `con_print()`, `contact_business_valid_email()`, `deserialize_int16()`, `deserialize_int32()`, `deserialize_int64()`, `deserialize_st()`, `deserialize_uint16()`, `deserialize_uint32()`, `deserialize_uint64()`, `http_body()`, `http_data_value_decode()`, `imap_build_array()`, `imap_build_array_isliteral()`, `imap_command_parser()`, `imap_fetch_body()`, `imap_fetch_body_portion()`, `imap_fetch_bodystructure()`, `imap_parse_address_breaker()`, `imap_parse_address_part()`, `imap_valid_sequence()`, `int16_digits()`, `int32_digits()`, `int64_digits()`, `int8_digits()`, `mail_add_required_headers()`, `mail_build_signature()`, `mail_discover_insertion_point()`, `mail_extract_address()`, `mail_get_boundary()`, `mail_get_chunk()`, `mail_header_end()`, `mail_headers()`, `mail_insert_chunk_base64()`, `mail_load_message_top()`, `mail_message()`, `mail_message_cleanup()`, `mail_mime_boundary()`, `mail_mime_child()`, `mail_mime_count()`, `mail_mime_header()`, `mail_mime_type_parameters_key()`, `mail_mime_type_parameters_value()`, `mail_modify_part()`, `mail_path_finder()`, `molten_stats()`, `naked_message_set()`, `ns_dupe()`, `org_key_get()`, `org_signet_get()`, `pop_num_parse()`, `pop_pass_parse()`, `pop_top_parse()`, `pop_user_parse()`, `prime_header_length()`, `prime_header_write()`, `prime_pem_wrap()`, `register_business_validate_password()`, `register_business_validate_username()`, `res_bind_create()`, `res_bind_free()`, `res_field_string()`, `res_row_store()`, `serialize_int16()`, `serialize_int32()`, `serialize_int64()`, `serialize_st()`, `serialize_uint16()`, `serialize_uint32()`, `serialize_uint64()`, `smtp_check_filters()`, `smtp_parse_auth()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, `smtp_parse_rcpt_to()`, `st_merge_opts()`, `st_realloc()`, `st_vaprint_opts()`, `st_vsprint()`, `stamp_counter_check()`, `tls_print()`, `tran_commit()`, `tran_rollback()`, `tran_start()`, `tree_truncate()`, `uint16_digits()`, `uint32_digits()`, `uint64_digits()`, `uint8_digits()`, `user_key_get()`, `user_request_get()`, and `user_signet_get()`.

5.50.3.8 size_t length_true

Definition at line 26 of file secure.c.

5.50.3.9 pthread_mutex_t lock

Definition at line 27 of file secure.c.

Referenced by `__attribute__()`, `lock_get()`, and `lock_release()`.

5.50.3.10 secured_t

Definition at line 18 of file secure.c.

Referenced by `mm_sec_alloc()`, `mm_sec_chunk_merge()`, `mm_sec_chunk_new()`, `mm_sec_chunk_next()`, `mm_sec_chunk_prev()`, `mm_sec_free()`, `mm_sec_realloc()`, and `mm_sec_start()`.

5.50.3.11 struct { ... } slab

Referenced by `mm_sec_secured()`.

5.51 src/core/parsers/case.c File Reference

```
#include "magma.h"
```

Functions

- [uchr_t upper_chr](#) ([uchr_t](#) *c*)
Return the uppercase representation of a character.
- [uchr_t lower_chr](#) ([uchr_t](#) *c*)
Return the lowercase representation of a character.
- [stringer_t * upper_st](#) ([stringer_t](#) **s*)
Transform a managed string (in-place) into uppercase.
- [stringer_t * lower_st](#) ([stringer_t](#) **s*)
Transform a managed string (in-place) into lowercase.

5.51.1 Function Documentation

5.51.1.1 [uchr_t lower_chr](#) ([uchr_t](#) *c*)

Return the lowercase representation of a character.

Parameters:

c the character to be transformed.

Returns:

the input character, in lowercase.

Definition at line 25 of file case.c.

Referenced by [auth_sanitize_address\(\)](#), [hex_decode_chr\(\)](#), [mm_cmp_ci_eq\(\)](#), [pop_user_parse\(\)](#), [smtp_parse_helo_domain\(\)](#), [smtp_parse_mail_from_path\(\)](#), [smtp_parse_rcpt_to\(\)](#), [st_cmp_ci_ends\(\)](#), [st_cmp_ci_eq\(\)](#), [st_cmp_ci_starts\(\)](#), and [st_search_ci\(\)](#).

5.51.1.2 [stringer_t * lower_st](#) ([stringer_t](#) * *s*)

Transform a managed string (in-place) into lowercase.

Parameters:

s the managed string to be modified.

Returns:

NULL on error, or a pointer to the input managed string, in lowercase.

Definition at line 58 of file case.c.

References [log_pedantic](#), and [st_empty_out\(\)](#).

Referenced by [contact_business_valid_email\(\)](#), [http_parse_header\(\)](#), [register_business_step1\(\)](#), and [smtp_rcpt_to\(\)](#).

5.51.1.3 `uchr_t upper_chr (uchr_t c)`

Return the uppercase representation of a character.

Parameters:

`c` the character to be transformed.

Returns:

the input character, in uppercase.

Definition at line 15 of file case.c.

5.51.1.4 `stringer_t* upper_st (stringer_t * s)`

Transform a managed string (in-place) into uppercase.

Parameters:

`s` the managed string to be modified.

Returns:

NULL on error, or a pointer to the input managed string, in uppercase.

Definition at line 35 of file case.c.

References `log_pedantic`, and `st_empty_out()`.

Referenced by `http_response_allow_cross()`, `imap_fetch_body_tag()`, `imap_fetch_bodystructure()`, and `mail_mime_type_parameters()`.

5.52 src/core/parsers/formats/formats.h File Reference

Data Structures

- struct [nvp_t](#)

Functions

- [nvp_t * nvp_alloc \(\)](#)
Allocate a new name/value pair and initialize it with the default settings.
- [void nvp_free \(nvp_t *nvp\)](#)
Free a name/value pair object from memory.
- [void nvp_init \(nvp_t *nvp\)](#)
- [int nvp_parse \(nvp_t *nvp, stringer_t *data\)](#)

5.52.1 Function Documentation

5.52.1.1 [nvp_t* nvp_alloc \(\)](#)

Allocate a new name/value pair and initialize it with the default settings.

Note:

Defaults use "\n" for a line separator, "#" for a comment starting character, and "=" as the assignment character.

Returns:

NULL on failure, or a pointer to the newly allocated name/value pair object.

Definition at line 73 of file [nvp.c](#).

References [inx_alloc\(\)](#), [log_info](#), [M_INX_HASHED](#), [mm_alloc\(\)](#), [mm_free\(\)](#), [nvp_t::pairs](#), [st_free\(\)](#), and [nvp_t::tokens](#).

Referenced by [config_load_cmdline_settings\(\)](#), and [config_load_file_settings\(\)](#).

5.52.1.2 [void nvp_free \(nvp_t * nvp\)](#)

Free a name/value pair object from memory.

Parameters:

nvp the name/value pair object to be freed.

Returns:

This function returns no value.

Definition at line 100 of file [nvp.c](#).

References [inx_free\(\)](#), [mm_free\(\)](#), and [nvp_t::pairs](#).

Referenced by [config_load_cmdline_settings\(\)](#), and [config_load_file_settings\(\)](#).

5.52.1.3 void nvp_init (nvp_t * *nvp*)**5.52.1.4 int nvp_parse (nvp_t * *nvp*, stringer_t * *data*)**

on success the number of valid pairs is returned on error -1 is returned

Definition at line 14 of file nvp.c.

References count, inx_insert(), log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_STRINGER, mt_set_type(), nvp_t::pairs, pl_char_get(), pl_data_get(), pl_empty(), pl_length_get(), pl_null(), pl_starts_with_char(), pl_trim(), placer_t, multi_t::st, st_free(), st_import(), st_length_get(), tok_get_bl(), tok_pop(), tok_pop_init_st(), nvp_t::tokens, and multi_t::val.

Referenced by config_load_cmdline_settings(), and config_load_file_settings().

5.53 src/core/parsers/formats/nvp.c File Reference

```
#include "magma.h"
```

Functions

- int [nvp_parse](#) ([nvp_t](#) *nvp, [stringer_t](#) *data)
- [nvp_t](#) * [nvp_alloc](#) ()
Allocate a new name/value pair and initialize it with the default settings.
- void [nvp_free](#) ([nvp_t](#) *nvp)
Free a name/value pair object from memory.

5.53.1 Function Documentation

5.53.1.1 [nvp_t](#)* [nvp_alloc](#) ()

Allocate a new name/value pair and initialize it with the default settings.

Note:

Defaults use "\n" for a line separator, "#" for a comment starting character, and "=" as the assignment character.

Returns:

NULL on failure, or a pointer to the newly allocated name/value pair object.

Definition at line 73 of file nvp.c.

References [inx_alloc\(\)](#), [log_info](#), [M_INX_HASHED](#), [mm_alloc\(\)](#), [mm_free\(\)](#), [nvp_t::pairs](#), [st_free\(\)](#), and [nvp_t::tokens](#).

Referenced by [config_load_cmdline_settings\(\)](#), and [config_load_file_settings\(\)](#).

5.53.1.2 void [nvp_free](#) ([nvp_t](#) * *nvp*)

Free a name/value pair object from memory.

Parameters:

nvp the name/value pair object to be freed.

Returns:

This function returns no value.

Definition at line 100 of file nvp.c.

References [inx_free\(\)](#), [mm_free\(\)](#), and [nvp_t::pairs](#).

Referenced by [config_load_cmdline_settings\(\)](#), and [config_load_file_settings\(\)](#).

5.53.1.3 int [nvp_parse](#) ([nvp_t](#) * *nvp*, [stringer_t](#) * *data*)

on success the number of valid pairs is returned on error -1 is returned

Definition at line 14 of file nvp.c.

References count, `inx_insert()`, `log_options`, `M_LOG_PEDANTIC`, `M_LOG_STACK_TRACE`, `M_TYPE_STRINGER`, `mt_set_type()`, `nvp_t::pairs`, `pl_char_get()`, `pl_data_get()`, `pl_empty()`, `pl_length_get()`, `pl_null()`, `pl_starts_with_char()`, `pl_trim()`, `placer_t`, `multi_t::st`, `st_free()`, `st_import()`, `st_length_get()`, `tok_get_bl()`, `tok_pop()`, `tok_pop_init_st()`, `nvp_t::tokens`, and `multi_t::val`.

Referenced by `config_load_cmdline_settings()`, and `config_load_file_settings()`.

5.54 src/core/parsers/line.c File Reference

```
#include "magma.h"
```

Functions

- [placer_t line_pl_bl](#) (char *block, size_t length, uint64_t number)
Get a placer pointing to the specified line (' delimited) of content in a data buffer.
- [placer_t line_pl_ns](#) (char *string, uint64_t number)
Get a placer pointing to the specified line (' delimited) of a null-terminated string.
- [placer_t line_pl_st](#) (stringer_t *string, uint64_t number)
Get a placer pointing to the specified line (' delimited) of a managed string.
- [placer_t line_pl_pl](#) (placer_t string, uint64_t number)
Get a placer pointing to the specified line (' delimited) of another placer.

5.54.1 Function Documentation

5.54.1.1 [placer_t line_pl_bl](#) (char * *block*, size_t *length*, uint64_t *number*)

Get a placer pointing to the specified line ('
' delimited) of content in a data buffer.

Parameters:

- block*** a pointer to the block of data to be scanned.
- length*** the length, in bytes, of the specified data buffer.
- number*** the zero-based index of the line to be retrieved from the buffer.

Returns:

a null placer on failure, or a placer pointing to the specified line on success.

Definition at line 17 of file line.c.

References [log_pedantic](#), [mm_empty\(\)](#), [pl_init\(\)](#), and [pl_null\(\)](#).

Referenced by [line_pl_ns\(\)](#), [line_pl_pl\(\)](#), and [line_pl_st\(\)](#).

5.54.1.2 [placer_t line_pl_ns](#) (char * *string*, uint64_t *number*)

Get a placer pointing to the specified line ('
' delimited) of a null-terminated string.

See also:

[line_pl_bl\(\)](#)

Parameters:

- string* a pointer to a null-terminated string to be scanned.
- number* the zero-based index of the line to be retrieved from the string.

Returns:

a null placer on failure, or a placer pointing to the specified line on success.

Definition at line 60 of file line.c.

References line_pl_bl(), and ns_length_get().

5.54.1.3 placer_t line_pl_pl (placer_t *string*, uint64_t *number*)

Get a placer pointing to the specified line (‘
’ delimited) of another placer.

Parameters:

- string* a pointer to the placer to be scanned.
- number* the zero-based index of the line to be retrieved from the placer.

Returns:

a null placer on failure, or a placer pointing to the specified line on success.

Definition at line 82 of file line.c.

References line_pl_bl(), pl_char_get(), and pl_length_get().

5.54.1.4 placer_t line_pl_st (stringer_t * *string*, uint64_t *number*)

Get a placer pointing to the specified line (‘
’ delimited) of a managed string.

Parameters:

- string* a pointer to the managed string to be scanned.
- number* the zero-based index of the line to be retrieved from the managed string.

Returns:

a null placer on failure, or a placer pointing to the specified line on success.

Definition at line 71 of file line.c.

References line_pl_bl(), st_char_get(), and st_length_get().

Referenced by client_read_line(), con_read_line(), and imap_parse_literal().

5.55 src/core/parsers/numbers/clamp.c File Reference

```
#include "magma.h"
```

Functions

- `uint8_t uint8_clamp` (`uint8_t min`, `uint8_t max`, `uint8_t number`)
Ensure a number is between the min and max values, otherwise return the boundary value.
- `uint16_t uint16_clamp` (`uint16_t min`, `uint16_t max`, `uint16_t number`)
Ensure a number is between the min and max values, otherwise return the boundary value.
- `uint32_t uint32_clamp` (`uint32_t min`, `uint32_t max`, `uint32_t number`)
Ensure a number is between the min and max values, otherwise return the boundary value.
- `uint64_t uint64_clamp` (`uint64_t min`, `uint64_t max`, `uint64_t number`)
Ensure a number is between the min and max values, otherwise return the boundary value.
- `int8_t int8_clamp` (`int8_t min`, `int8_t max`, `int8_t number`)
Ensure a number is between the min and max values, otherwise return the boundary value.
- `int16_t int16_clamp` (`int16_t min`, `int16_t max`, `int16_t number`)
Ensure a number is between the min and max values, otherwise return the boundary value.
- `int32_t int32_clamp` (`int32_t min`, `int32_t max`, `int32_t number`)
Ensure a number is between the min and max values, otherwise return the boundary value.
- `int64_t int64_clamp` (`int64_t min`, `int64_t max`, `int64_t number`)
Ensure a number is between the min and max values, otherwise return the boundary value.

5.55.1 Function Documentation

5.55.1.1 `int16_t int16_clamp` (`int16_t min`, `int16_t max`, `int16_t number`)

Ensure a number is between the min and max values, otherwise return the boundary value. [clamp.c](#)

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 125 of file `clamp.c`.

References `log_pedantic`.

5.55.1.2 int32_t int32_clamp (int32_t min, int32_t max, int32_t number)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 146 of file clamp.c.

References log_pedantic.

5.55.1.3 int64_t int64_clamp (int64_t min, int64_t max, int64_t number)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 167 of file clamp.c.

References log_pedantic.

5.55.1.4 int8_t int8_clamp (int8_t min, int8_t max, int8_t number)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 104 of file clamp.c.

References log_pedantic.

5.55.1.5 uint16_t uint16_clamp (uint16_t min, uint16_t max, uint16_t number)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 41 of file clamp.c.

References log_pedantic.

5.55.1.6 uint32_t uint32_clamp (uint32_t min, uint32_t max, uint32_t number)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 62 of file clamp.c.

References log_pedantic.

5.55.1.7 uint64_t uint64_clamp (uint64_t min, uint64_t max, uint64_t number)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 83 of file clamp.c.

References log_pedantic.

Referenced by stacie_derive_rounds(), and stmt_exec_affected_conn().

5.55.1.8 uint8_t uint8_clamp (uint8_t *min*, uint8_t *max*, uint8_t *number*)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 20 of file clamp.c.

References [log_pedantic](#).

5.56 src/core/parsers/numbers/digits.c File Reference

```
#include "magma.h"
```

Functions

- `size_t uint64_digits (uint64_t number)`
Count the number of digits needed to represent a base-10 64-bit unsigned integer.
- `size_t uint32_digits (uint32_t number)`
Count the number of digits needed to represent a base-10 32-bit unsigned integer.
- `size_t uint16_digits (uint16_t number)`
Count the number of digits needed to represent a base-10 16-bit unsigned integer.
- `size_t uint8_digits (uint8_t number)`
Count the number of digits needed to represent a base-10 8-bit unsigned integer.
- `size_t int64_digits (int64_t number)`
Count the number of digits needed to represent a base-10 64-bit signed integer.
- `size_t int32_digits (int32_t number)`
Count the number of digits needed to represent a base-10 32-bit signed integer.
- `size_t int16_digits (int16_t number)`
Count the number of digits needed to represent a base-10 16-bit signed integer.
- `size_t int8_digits (int8_t number)`
Count the number of digits needed to represent a base-10 8-bit signed integer.

5.56.1 Function Documentation

5.56.1.1 `size_t int16_digits (int16_t number)`

Count the number of digits needed to represent a base-10 16-bit signed integer. [digits.c](#)

Returns:

the number of digits needed to represent the specified integer, including a negative sign.

Definition at line 112 of file `digits.c`.

References `length`.

Referenced by `serialize_int16()`.

5.56.1.2 `size_t int32_digits (int32_t number)`

Count the number of digits needed to represent a base-10 32-bit signed integer.

Returns:

the number of digits needed to represent the specified integer, including a negative sign.

Definition at line 93 of file digits.c.

References length.

Referenced by serialize_int32().

5.56.1.3 size_t int64_digits (int64_t *number*)

Count the number of digits needed to represent a base-10 64-bit signed integer.

Returns:

the number of digits needed to represent the specified integer, including a negative sign.

Definition at line 74 of file digits.c.

References length.

Referenced by serialize_int64().

5.56.1.4 size_t int8_digits (int8_t *number*)

Count the number of digits needed to represent a base-10 8-bit signed integer.

Returns:

the number of digits needed to represent the specified integer, including a negative sign.

Definition at line 131 of file digits.c.

References length.

5.56.1.5 size_t uint16_digits (uint16_t *number*)

Count the number of digits needed to represent a base-10 16-bit unsigned integer.

Returns:

the number of digits needed to represent the specified integer.

Definition at line 44 of file digits.c.

References length.

Referenced by serialize_uint16().

5.56.1.6 size_t uint32_digits (uint32_t *number*)

Count the number of digits needed to represent a base-10 32-bit unsigned integer.

Returns:

the number of digits needed to represent the specified integer.

Definition at line 29 of file digits.c.

References length.

Referenced by serialize_uint32().

5.56.1.7 `size_t uint64_digits (uint64_t number)`

Count the number of digits needed to represent a base-10 64-bit unsigned integer.

Returns:

the number of digits needed to represent specified integer.

Definition at line 14 of file digits.c.

References length.

Referenced by mail_build_signature(), serialize_uint64(), sess_key(), and smtp_store_spamsig().

5.56.1.8 `size_t uint8_digits (uint8_t number)`

Count the number of digits needed to represent a base-10 8-bit unsigned integer.

Returns:

the number of digits needed to represent the specified integer.

Definition at line 59 of file digits.c.

References length.

5.57 src/core/parsers/numbers/numbers.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t float_conv](#) ([stringer_t](#) *s, [float_t](#) *number)
Convert a managed string to a float.
- [bool_t double_conv](#) ([stringer_t](#) *s, [double_t](#) *number)
Convert a managed string to a double.
- [bool_t size_conv_bl](#) (void *block, [size_t](#) length, [size_t](#) *number)
Convert a numerical string into a size_t value.
- [bool_t ssize_conv_bl](#) (void *block, [size_t](#) length, [ssize_t](#) *number)
Convert a numerical string into an ssize_t value.
- [bool_t uint64_conv_bl](#) (void *block, [size_t](#) length, [uint64_t](#) *number)
Convert a numerical string into an unsigned 64-bit integer.
- [bool_t uint64_conv_ns](#) (char *string, [uint64_t](#) *number)
Convert a null-terminated string into an unsigned 64-bit integer.
- [bool_t uint64_conv_st](#) ([stringer_t](#) *string, [uint64_t](#) *number)
Convert a managed string into an unsigned 64-bit integer.
- [bool_t uint64_conv_pl](#) ([placer_t](#) string, [uint64_t](#) *number)
Convert a placer into an unsigned 64-bit integer.
- [bool_t uint32_conv_bl](#) (void *block, [size_t](#) length, [uint32_t](#) *number)
Convert a numerical string into an unsigned 32-bit integer.
- [bool_t uint32_conv_ns](#) (char *string, [uint32_t](#) *number)
Convert a null-terminated string into an unsigned 32-bit integer.
- [bool_t uint32_conv_st](#) ([stringer_t](#) *string, [uint32_t](#) *number)
Convert a managed string into an unsigned 32-bit integer.
- [bool_t uint16_conv_bl](#) (void *block, [size_t](#) length, [uint16_t](#) *number)
Convert a numerical string to a 16-bit unsigned integer.
- [bool_t uint16_conv_ns](#) (char *string, [uint16_t](#) *number)
Convert a numerical string to a 16-bit unsigned integer.
- [bool_t uint16_conv_st](#) ([stringer_t](#) *string, [uint16_t](#) *number)
Convert a numerical string to a 16-bit unsigned integer.
- [bool_t uint8_conv_bl](#) (void *block, [size_t](#) length, [uint8_t](#) *number)
Convert a numerical string to an unsigned 8-bit unsigned integer.
- [bool_t uint8_conv_ns](#) (char *string, [uint8_t](#) *number)

Convert a numerical string to an 8-bit unsigned integer.

- `bool_t uint8_conv_st (stringer_t *string, uint8_t *number)`

Convert a numerical string to an 8-bit unsigned integer.

- `bool_t int64_conv_bl (void *block, size_t length, int64_t *number)`

Convert a numerical string into a signed 64-bit integer.

- `bool_t int64_conv_ns (char *string, int64_t *number)`

Convert a null-terminated string into a signed 64-bit integer.

- `bool_t int64_conv_st (stringer_t *string, int64_t *number)`

Convert a managed string into a signed 64-bit integer.

- `bool_t int32_conv_bl (void *block, size_t length, int32_t *number)`

Convert a numerical string into a signed 32-bit integer.

- `bool_t int32_conv_ns (char *string, int32_t *number)`

Convert a null-terminated string into a signed 32-bit integer.

- `bool_t int32_conv_st (stringer_t *string, int32_t *number)`

Convert a managed string into a signed 32-bit integer.

- `bool_t int16_conv_bl (void *block, size_t length, int16_t *number)`

Convert a numerical string into a signed 16-bit integer.

- `bool_t int16_conv_ns (char *string, int16_t *number)`

Convert a null-terminated string into a signed 16-bit integer.

- `bool_t int16_conv_st (stringer_t *string, int16_t *number)`

Convert a managed string into a signed 16-bit integer.

- `bool_t int8_conv_bl (void *block, size_t length, int8_t *number)`

Convert a numerical string into a signed 8-bit integer.

- `bool_t int8_conv_ns (char *string, int8_t *number)`

Convert a null-terminated string into a signed 8-bit integer.

- `bool_t int8_conv_st (stringer_t *string, int8_t *number)`

Convert a managed string into a signed 8-bit integer.

- `stringer_t * uint32_put_no (uint32_t val)`

Create a stringer, storing a 4-byte value in network byte order.

- `stringer_t * uint24_put_no (uint32_t val)`

Create a stringer, storing a 3-byte value in network byte order.

- `stringer_t * uint16_put_no (uint16_t val)`

Create a stringer, storing a 2-byte value in network byte order.

- `uint32_t uint32_get_no (stringer_t *s)`

Fetch a 4 byte network order value from a stringer and return it in host byte order.

- `uint32_t uint24_get_no (stringer_t *s)`

Fetch a 3 byte network order value from a stringer and return it in host byte order.

- `uint16_t uint16_get_no (stringer_t *s)`

Fetch a 2 byte network order value from a stringer and return it in host byte order.

5.57.1 Function Documentation

5.57.1.1 `bool_t double_conv (stringer_t *s, double_t *number)`

Convert a managed string to a double. [numbers.c](#)

Parameters:

s the managed string to be converted.

number a pointer to a double where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 48 of file numbers.c.

References `log_pedantic`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

5.57.1.2 `bool_t float_conv (stringer_t *s, float_t *number)`

Convert a managed string to a float.

Parameters:

s the managed string to be converted.

number a pointer to a float where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 16 of file numbers.c.

References `log_pedantic`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

5.57.1.3 `bool_t int16_conv_bl (void *block, size_t length, int16_t *number)`

Convert a numerical string into a signed 16-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to a signed 16-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 559 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_int16(), int16_conv_ns(), and int16_conv_st().

5.57.1.4 bool_t int16_conv_ns (char * *string*, int16_t * *number*)

Convert a null-terminated string into a signed 16-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to a signed 16-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 615 of file numbers.c.

References int16_conv_bl(), and ns_length_get().

5.57.1.5 bool_t int16_conv_st (stringer_t * *string*, int16_t * *number*)

Convert a managed string into a signed 16-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to a signed 16-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 626 of file numbers.c.

References int16_conv_bl(), st_data_get(), and st_length_get().

Referenced by xml_get_xpath_int16().

5.57.1.6 bool_t int32_conv_bl (void * *block*, size_t *length*, int32_t * *number*)

Convert a numerical string into a signed 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.
length the length, in bytes, of the numerical data.
number a pointer to a signed 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 480 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_int32(), http_parse_header(), imap_fetch_parse_partial(), int32_conv_ns(), and int32_conv_st().

5.57.1.7 bool_t int32_conv_ns (char * *string*, int32_t * *number*)

Convert a null-terminated string into a signed 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.
number a pointer to a signed 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 536 of file numbers.c.

References int32_conv_bl(), and ns_length_get().

Referenced by process_find_pid(), and process_kill().

5.57.1.8 bool_t int32_conv_st (stringer_t * *string*, int32_t * *number*)

Convert a managed string into a signed 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.
number a pointer to a signed 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 547 of file numbers.c.

References int32_conv_bl(), st_data_get(), and st_length_get().

Referenced by cache_set_value(), config_value_set(), relay_set_value(), servers_set_value(), and xml_get_xpath_int32().

5.57.1.9 `bool_t int64_conv_bl (void * block, size_t length, int64_t * number)`

Convert a numerical string into a signed 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to a signed 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 401 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_int64(), int64_conv_ns(), int64_conv_st(), and ssize_conv_bl().

5.57.1.10 `bool_t int64_conv_ns (char * string, int64_t * number)`

Convert a null-terminated string into a signed 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to a signed 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 457 of file numbers.c.

References int64_conv_bl(), and ns_length_get().

5.57.1.11 `bool_t int64_conv_st (stringer_t * string, int64_t * number)`

Convert a managed string into a signed 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to a signed 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 468 of file numbers.c.

References `int64_conv_bl()`, `st_data_get()`, and `st_length_get()`.

Referenced by `cache_set_value()`, `config_value_set()`, `relay_set_value()`, `servers_set_value()`, and `xml_get_xpath_int64()`.

5.57.1.12 `bool_t int8_conv_bl (void * block, size_t length, int8_t * number)`

Convert a numerical string into a signed 8-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to a signed 8-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 638 of file numbers.c.

References `data`, and `log_pedantic`.

Referenced by `int8_conv_ns()`, and `int8_conv_st()`.

5.57.1.13 `bool_t int8_conv_ns (char * string, int8_t * number)`

Convert a null-terminated string into a signed 8-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to a signed 8-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 694 of file numbers.c.

References `int8_conv_bl()`, and `ns_length_get()`.

5.57.1.14 `bool_t int8_conv_st (stringer_t * string, int8_t * number)`

Convert a managed string into a signed 8-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to a signed 8-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 705 of file numbers.c.

References `int8_conv_bl()`, `st_data_get()`, and `st_length_get()`.

Referenced by `cache_set_value()`, `config_value_set()`, `relay_set_value()`, `servers_set_value()`, and `xml_get_xpath_int8()`.

5.57.1.15 bool_t size_conv_bl (void * block, size_t length, size_t * number)

Convert a numerical string into a size_t value.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to an size_t value where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 82 of file numbers.c.

References `uint64_conv_bl()`.

Referenced by `http_body()`.

5.57.1.16 bool_t ssize_conv_bl (void * block, size_t length, ssize_t * number)

Convert a numerical string into an ssize_t value.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to an ssize_t value where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 94 of file numbers.c.

References `int64_conv_bl()`.

5.57.1.17 bool_t uint16_conv_bl (void * *block*, size_t *length*, uint16_t * *number*)

Convert a numerical string to a 16-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the block of memory to be converted.

length the length, in bytes, of the numerical string.

number a pointer to an unsigned 16-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 259 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_uint16(), uint16_conv_ns(), and uint16_conv_st().

5.57.1.18 bool_t uint16_conv_ns (char * *string*, uint16_t * *number*)

Convert a numerical string to a 16-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to a null-terminated string containing the data to be converted.

number a pointer to an unsigned 16-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 307 of file numbers.c.

References ns_length_get(), and uint16_conv_bl().

5.57.1.19 bool_t uint16_conv_st (stringer_t * *string*, uint16_t * *number*)

Convert a numerical string to a 16-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to a managed string containing the data to be converted.

number a pointer to an unsigned 16-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 318 of file numbers.c.

References `st_data_get()`, `st_length_get()`, and `uint16_conv_bl()`.

Referenced by `cache_set_value()`, `config_value_set()`, `relay_set_value()`, `servers_set_value()`, and `xml_get_xpath_uint16()`.

5.57.1.20 `uint16_t uint16_get_no (stringer_t * s)`

Fetch a 2 byte network order value from a stringer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 2 network order bytes in the buffer in host byte order. 0 on failure.

Definition at line 852 of file numbers.c.

References `data`, `log_pedantic`, `st_data_get()`, `st_empty`, and `st_length_get()`.

5.57.1.21 `stringer_t* uint16_put_no (uint16_t val)`

Create a stringer, storing a 2-byte value in network byte order.

Parameters:

s a pointer to the data buffer where the value will be inserted.

val the 2 byte value to be placed in the specified buffer, in network byte order.

Returns:

Pointer to stringer on success, NULL on failure.

Definition at line 774 of file numbers.c.

References `data`, `log_pedantic`, `st_alloc()`, `st_data_get()`, and `st_length_set()`.

5.57.1.22 `uint32_t uint24_get_no (stringer_t * s)`

Fetch a 3 byte network order value from a stringer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 3 network order bytes in the buffer in host byte order. 0 on failure.

Definition at line 827 of file numbers.c.

References `data`, `log_pedantic`, `st_data_get()`, `st_empty`, and `st_length_get()`.

5.57.1.23 `stringer_t* uint24_put_no (uint32_t val)`

Create a stringer, storing a 3-byte value in network byte order.

Parameters:

- s* a pointer to the data buffer where the value will be inserted.
- val* the 3 byte value to be placed in the specified buffer, in network byte order.

Returns:

Pointer to stringer on success, NULL on failure.

Definition at line 745 of file numbers.c.

References data, log_pedantic, st_alloc(), st_data_get(), and st_length_set().

5.57.1.24 bool_t uint32_conv_bl (void * *block*, size_t *length*, uint32_t * *number*)

Convert a numerical string into an unsigned 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

- block* a pointer to the numerical string to be converted.
- length* the length, in bytes, of the numerical data.
- number* a pointer to an unsigned 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 188 of file numbers.c.

References data, and log_pedantic.

Referenced by cache_config(), deserialize_uint32(), relay_config(), servers_config(), uint32_conv_ns(), and uint32_conv_st().

5.57.1.25 bool_t uint32_conv_ns (char * *string*, uint32_t * *number*)

Convert a null-terminated string into an unsigned 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

- string* a pointer to the null-terminated string to be converted.
- number* a pointer to an unsigned 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 236 of file numbers.c.

References ns_length_get(), and uint32_conv_bl().

5.57.1.26 bool_t uint32_conv_st (stringer_t * *string*, uint32_t * *number*)

Convert a managed string into an unsigned 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to an unsigned 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 247 of file numbers.c.

References st_data_get(), st_length_get(), and uint32_conv_bl().

Referenced by cache_set_value(), config_value_set(), imap_fetch_body_part(), imap_search_messages_date_compare(), relay_set_value(), servers_set_value(), and xml_get_xpath_uint32().

5.57.1.27 uint32_t uint32_get_no (stringer_t * *s*)

Fetch a 4 byte network order value from a stringer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 4 network order bytes in the buffer in host byte order. 0 on failure.

Definition at line 801 of file numbers.c.

References data, log_pedantic, st_data_get(), st_empty, and st_length_get().

5.57.1.28 stringer_t* uint32_put_no (uint32_t *val*)

Create a stringer, storing a 4-byte value in network byte order.

Parameters:

s a pointer to the data buffer where the value will be inserted.

val the 4 byte value to be placed in the specified buffer, in network byte order.

Returns:

Pointer to stringer on success, NULL on failure.

Definition at line 715 of file numbers.c.

References data, log_pedantic, st_alloc(), st_data_get(), and st_length_set().

5.57.1.29 bool_t uint64_conv_bl (void * *block*, size_t *length*, uint64_t * *number*)

Convert a numerical string into an unsigned 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to an unsigned 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 106 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_uint64(), imap_parse_literal(), pop_num_parse(), pop_top_parse(), size_conv_bl(), smtp_check_receive_quota(), uint64_conv_ns(), uint64_conv_pl(), and uint64_conv_st().

5.57.1.30 bool_t uint64_conv_ns (char * *string*, uint64_t * *number*)

Convert a null-terminated string into an unsigned 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to an unsigned 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 154 of file numbers.c.

References ns_length_get(), and uint64_conv_bl().

Referenced by json_api_dispatch(), lib_load_xml(), and portal_endpoint().

5.57.1.31 bool_t uint64_conv_pl (placer_t *string*, uint64_t * *number*)

Convert a placer into an unsigned 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the placer to be converted.

number a pointer to an unsigned 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 176 of file numbers.c.

References `pl_data_get()`, `pl_length_get()`, and `uint64_conv_bl()`.

Referenced by `build_version_major()`, `build_version_minor()`, `build_version_patch()`, `smtp_parse_mail_from_path()`, and `stamp_counter_check()`.

5.57.1.32 bool_t uint64_conv_st (stringer_t * string, uint64_t * number)

Convert a managed string into an unsigned 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to an unsigned 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 165 of file numbers.c.

References `st_data_get()`, `st_length_get()`, and `uint64_conv_bl()`.

Referenced by `cache_set_value()`, `config_value_set()`, `imap_narrow_messages()`, `imap_search_messages_range()`, `imap_search_messages_size()`, `portal_get_upload_attachment()`, `relay_set_value()`, `servers_set_value()`, `teacher_process()`, and `xml_get_xpath_uint64()`.

5.57.1.33 bool_t uint8_conv_bl (void * block, size_t length, uint8_t * number)

Convert a numerical string to an unsigned 8-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the block of memory to be converted.

length the length, in bytes, of the numerical string.

number a pointer to an unsigned 8-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 330 of file numbers.c.

References `data`, and `log_pedantic`.

Referenced by `ip_subnet_st()`, `uint8_conv_ns()`, and `uint8_conv_st()`.

5.57.1.34 bool_t uint8_conv_ns (char * *string*, uint8_t * *number*)

Convert a numerical string to an 8-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to a null-terminated string containing the data to be converted.

number a pointer to an unsigned 8-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 378 of file numbers.c.

References ns_length_get(), and uint8_conv_bl().

5.57.1.35 bool_t uint8_conv_st (stringer_t * *string*, uint8_t * *number*)

Convert a numerical string to an 8-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to a managed string containing the data to be converted.

number a pointer to an unsigned 8-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 389 of file numbers.c.

References st_data_get(), st_length_get(), and uint8_conv_bl().

Referenced by cache_set_value(), config_value_set(), relay_set_value(), servers_set_value(), and xml_get_xpath_uint8().

5.58 src/core/parsers/numbers/numbers.h File Reference

Functions

- `size_t int16_digits (int16_t number)`
digits.c
- `size_t int32_digits (int32_t number)`
Count the number of digits needed to represent a base-10 32-bit signed integer.
- `size_t int64_digits (int64_t number)`
Count the number of digits needed to represent a base-10 64-bit signed integer.
- `size_t int8_digits (int8_t number)`
Count the number of digits needed to represent a base-10 8-bit signed integer.
- `size_t uint16_digits (uint16_t number)`
Count the number of digits needed to represent a base-10 16-bit unsigned integer.
- `size_t uint32_digits (uint32_t number)`
Count the number of digits needed to represent a base-10 32-bit unsigned integer.
- `size_t uint64_digits (uint64_t number)`
Count the number of digits needed to represent a base-10 64-bit unsigned integer.
- `size_t uint8_digits (uint8_t number)`
Count the number of digits needed to represent a base-10 8-bit unsigned integer.
- `int16_t int16_clamp (int16_t min, int16_t max, int16_t number)`
clamp.c
- `int32_t int32_clamp (int32_t min, int32_t max, int32_t number)`
Ensure a number is between the min and max values, otherwise return the boundary value.
- `int64_t int64_clamp (int64_t min, int64_t max, int64_t number)`
Ensure a number is between the min and max values, otherwise return the boundary value.
- `int8_t int8_clamp (int8_t min, int8_t max, int8_t number)`
Ensure a number is between the min and max values, otherwise return the boundary value.
- `uint16_t uint16_clamp (uint16_t min, uint16_t max, uint16_t number)`
Ensure a number is between the min and max values, otherwise return the boundary value.
- `uint32_t uint32_clamp (uint32_t min, uint32_t max, uint32_t number)`
Ensure a number is between the min and max values, otherwise return the boundary value.
- `uint64_t uint64_clamp (uint64_t min, uint64_t max, uint64_t number)`
Ensure a number is between the min and max values, otherwise return tuint_the boundary value.
- `uint8_t uint8_clamp (uint8_t min, uint8_t max, uint8_t number)`
Ensure a number is between the min and max values, otherwise return the boundary value.
- `bool_t double_conv (stringer_t *s, double_t *number)`

numbers.c

- `bool_t float_conv (stringer_t *s, float_t *number)`
Convert a managed string to a float.
- `bool_t int16_conv_bl (void *block, size_t length, int16_t *number)`
Convert a numerical string into a signed 16-bit integer.
- `bool_t int16_conv_ns (char *string, int16_t *number)`
Convert a null-terminated string into a signed 16-bit integer.
- `bool_t int16_conv_st (stringer_t *string, int16_t *number)`
Convert a managed string into a signed 16-bit integer.
- `bool_t int32_conv_bl (void *block, size_t length, int32_t *number)`
Convert a numerical string into a signed 32-bit integer.
- `bool_t int32_conv_ns (char *string, int32_t *number)`
Convert a null-terminated string into a signed 32-bit integer.
- `bool_t int32_conv_st (stringer_t *string, int32_t *number)`
Convert a managed string into a signed 32-bit integer.
- `bool_t int64_conv_bl (void *block, size_t length, int64_t *number)`
Convert a numerical string into a signed 64-bit integer.
- `bool_t int64_conv_ns (char *string, int64_t *number)`
Convert a null-terminated string into a signed 64-bit integer.
- `bool_t int64_conv_st (stringer_t *string, int64_t *number)`
Convert a managed string into a signed 64-bit integer.
- `bool_t int8_conv_bl (void *block, size_t length, int8_t *number)`
Convert a numerical string into a signed 8-bit integer.
- `bool_t int8_conv_ns (char *string, int8_t *number)`
Convert a null-terminated string into a signed 8-bit integer.
- `bool_t int8_conv_st (stringer_t *string, int8_t *number)`
Convert a managed string into a signed 8-bit integer.
- `bool_t size_conv_bl (void *block, size_t length, size_t *number)`
Convert a numerical string into a size_t value.
- `bool_t ssize_conv_bl (void *block, size_t length, ssize_t *number)`
Convert a numerical string into an ssize_t value.
- `bool_t uint16_conv_bl (void *block, size_t length, uint16_t *number)`
Convert a numerical string to a 16-bit unsigned integer.
- `bool_t uint16_conv_ns (char *string, uint16_t *number)`
Convert a numerical string to a 16-bit unsigned integer.

- `bool_t uint16_conv_st (stringer_t *string, uint16_t *number)`
Convert a numerical string to a 16-bit unsigned integer.
- `bool_t uint32_conv_bl (void *block, size_t length, uint32_t *number)`
Convert a numerical string into an unsigned 32-bit integer.
- `bool_t uint32_conv_ns (char *string, uint32_t *number)`
Convert a null-terminated string into an unsigned 32-bit integer.
- `bool_t uint32_conv_st (stringer_t *string, uint32_t *number)`
Convert a managed string into an unsigned 32-bit integer.
- `bool_t uint64_conv_bl (void *block, size_t length, uint64_t *number)`
Convert a numerical string into an unsigned 64-bit integer.
- `bool_t uint64_conv_ns (char *string, uint64_t *number)`
Convert a null-terminated string into an unsigned 64-bit integer.
- `bool_t uint64_conv_pl (placer_t string, uint64_t *number)`
Convert a placer into an unsigned 64-bit integer.
- `bool_t uint64_conv_st (stringer_t *string, uint64_t *number)`
Convert a managed string into an unsigned 64-bit integer.
- `bool_t uint8_conv_bl (void *block, size_t length, uint8_t *number)`
Convert a numerical string to an unsigned 8-bit unsigned integer.
- `bool_t uint8_conv_ns (char *string, uint8_t *number)`
Convert a numerical string to an 8-bit unsigned integer.
- `bool_t uint8_conv_st (stringer_t *string, uint8_t *number)`
Convert a numerical string to an 8-bit unsigned integer.
- `stringer_t * uint32_put_no (uint32_t val)`
Create a stringer, storing a 4-byte value in network byte order.
- `stringer_t * uint24_put_no (uint32_t val)`
Create a stringer, storing a 3-byte value in network byte order.
- `stringer_t * uint16_put_no (uint16_t val)`
Create a stringer, storing a 2-byte value in network byte order.
- `uint32_t uint32_get_no (stringer_t *s)`
Fetch a 4 byte network order value from a stringer and return it in host byte order.
- `uint32_t uint24_get_no (stringer_t *s)`
Fetch a 3 byte network order value from a stringer and return it in host byte order.
- `uint16_t uint16_get_no (stringer_t *s)`
Fetch a 2 byte network order value from a stringer and return it in host byte order.

5.58.1 Function Documentation

5.58.1.1 `bool_t double_conv (stringer_t * s, double_t * number)`

[numbers.c](#) [numbers.c](#)

Parameters:

- s* the managed string to be converted.
- number* a pointer to a double where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 48 of file numbers.c.

References `log_pedantic`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

5.58.1.2 `bool_t float_conv (stringer_t * s, float_t * number)`

Convert a managed string to a float.

Parameters:

- s* the managed string to be converted.
- number* a pointer to a float where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 16 of file numbers.c.

References `log_pedantic`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

5.58.1.3 `int16_t int16_clamp (int16_t min, int16_t max, int16_t number)`

[clamp.c](#) [clamp.c](#)

Parameters:

- min* minimum value for the clamp
- max* maximum value for the clamp
- number* number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 125 of file clamp.c.

References `log_pedantic`.

5.58.1.4 bool_t int16_conv_bl (void * *block*, size_t *length*, int16_t * *number*)

Convert a numerical string into a signed 16-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to a signed 16-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 559 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_int16(), int16_conv_ns(), and int16_conv_st().

5.58.1.5 bool_t int16_conv_ns (char * *string*, int16_t * *number*)

Convert a null-terminated string into a signed 16-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to a signed 16-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 615 of file numbers.c.

References int16_conv_bl(), and ns_length_get().

5.58.1.6 bool_t int16_conv_st (stringer_t * *string*, int16_t * *number*)

Convert a managed string into a signed 16-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to a signed 16-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 626 of file numbers.c.

References `int16_conv_bl()`, `st_data_get()`, and `st_length_get()`.

Referenced by `xml_get_xpath_int16()`.

5.58.1.7 `size_t int16_digits (int16_t number)`

[digits.c digits.c](#)

Returns:

the number of digits needed to represent the specified integer, including a negative sign.

Definition at line 112 of file digits.c.

References `length`.

Referenced by `serialize_int16()`.

5.58.1.8 `int32_t int32_clamp (int32_t min, int32_t max, int32_t number)`

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp

max maximum value for the clamp

number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 146 of file clamp.c.

References `log_pedantic`.

5.58.1.9 `bool_t int32_conv_bl (void * block, size_t length, int32_t * number)`

Convert a numerical string into a signed 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to a signed 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 480 of file numbers.c.

References `data`, and `log_pedantic`.

Referenced by `deserialize_int32()`, `http_parse_header()`, `imap_fetch_parse_partial()`, `int32_conv_ns()`, and `int32_conv_st()`.

5.58.1.10 bool_t int32_conv_ns (char * *string*, int32_t * *number*)

Convert a null-terminated string into a signed 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to a signed 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 536 of file numbers.c.

References int32_conv_bl(), and ns_length_get().

Referenced by process_find_pid(), and process_kill().

5.58.1.11 bool_t int32_conv_st (stringer_t * *string*, int32_t * *number*)

Convert a managed string into a signed 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to a signed 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 547 of file numbers.c.

References int32_conv_bl(), st_data_get(), and st_length_get().

Referenced by cache_set_value(), config_value_set(), relay_set_value(), servers_set_value(), and xml_get_xpath_int32().

5.58.1.12 size_t int32_digits (int32_t *number*)

Count the number of digits needed to represent a base-10 32-bit signed integer.

Returns:

the number of digits needed to represent the specified integer, including a negative sign.

Definition at line 93 of file digits.c.

References length.

Referenced by serialize_int32().

5.58.1.13 `int64_t int64_clamp(int64_t min, int64_t max, int64_t number)`

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 167 of file clamp.c.

References log_pedantic.

5.58.1.14 `bool_t int64_conv_bl(void *block, size_t length, int64_t *number)`

Convert a numerical string into a signed 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.
length the length, in bytes, of the numerical data.
number a pointer to a signed 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 401 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_int64(), int64_conv_ns(), int64_conv_st(), and ssize_conv_bl().

5.58.1.15 `bool_t int64_conv_ns(char *string, int64_t *number)`

Convert a null-terminated string into a signed 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.
number a pointer to a signed 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 457 of file numbers.c.

References int64_conv_bl(), and ns_length_get().

5.58.1.16 bool_t int64_conv_st (stringer_t * *string*, int64_t * *number*)

Convert a managed string into a signed 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to a signed 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 468 of file numbers.c.

References int64_conv_bl(), st_data_get(), and st_length_get().

Referenced by cache_set_value(), config_value_set(), relay_set_value(), servers_set_value(), and xml_get_xpath_int64().

5.58.1.17 size_t int64_digits (int64_t *number*)

Count the number of digits needed to represent a base-10 64-bit signed integer.

Returns:

the number of digits needed to represent the specified integer, including a negative sign.

Definition at line 74 of file digits.c.

References length.

Referenced by serialize_int64().

5.58.1.18 int8_t int8_clamp (int8_t *min*, int8_t *max*, int8_t *number*)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp

max maximum value for the clamp

number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 104 of file clamp.c.

References log_pedantic.

5.58.1.19 bool_t int8_conv_bl (void * *block*, size_t *length*, int8_t * *number*)

Convert a numerical string into a signed 8-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to a signed 8-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 638 of file numbers.c.

References data, and log_pedantic.

Referenced by int8_conv_ns(), and int8_conv_st().

5.58.1.20 bool_t int8_conv_ns (char * *string*, int8_t * *number*)

Convert a null-terminated string into a signed 8-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to a signed 8-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 694 of file numbers.c.

References int8_conv_bl(), and ns_length_get().

5.58.1.21 bool_t int8_conv_st (stringer_t * *string*, int8_t * *number*)

Convert a managed string into a signed 8-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to a signed 8-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 705 of file numbers.c.

References `int8_conv_bl()`, `st_data_get()`, and `st_length_get()`.

Referenced by `cache_set_value()`, `config_value_set()`, `relay_set_value()`, `servers_set_value()`, and `xml_get_xpath_int8()`.

5.58.1.22 `size_t int8_digits (int8_t number)`

Count the number of digits needed to represent a base-10 8-bit signed integer.

Returns:

the number of digits needed to represent the specified integer, including a negative sign.

Definition at line 131 of file digits.c.

References `length`.

5.58.1.23 `bool_t size_conv_bl (void * block, size_t length, size_t * number)`

Convert a numerical string into a `size_t` value.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to an `size_t` value where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 82 of file numbers.c.

References `uint64_conv_bl()`.

Referenced by `http_body()`.

5.58.1.24 `bool_t ssize_conv_bl (void * block, size_t length, ssize_t * number)`

Convert a numerical string into an `ssize_t` value.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to an `ssize_t` value where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 94 of file numbers.c.

References `int64_conv_bl()`.

5.58.1.25 uint16_t uint16_clamp (uint16_t *min*, uint16_t *max*, uint16_t *number*)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 41 of file clamp.c.

References log_pedantic.

5.58.1.26 bool_t uint16_conv_bl (void * *block*, size_t *length*, uint16_t * *number*)

Convert a numerical string to a 16-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the block of memory to be converted.
length the length, in bytes, of the numerical string.
number a pointer to an unsigned 16-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 259 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_uint16(), uint16_conv_ns(), and uint16_conv_st().

5.58.1.27 bool_t uint16_conv_ns (char * *string*, uint16_t * *number*)

Convert a numerical string to a 16-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to a null-terminated string containing the data to be converted.
number a pointer to an unsigned 16-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 307 of file numbers.c.

References ns_length_get(), and uint16_conv_bl().

5.58.1.28 bool_t uint16_conv_st (stringer_t * *string*, uint16_t * *number*)

Convert a numerical string to a 16-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to a managed string containing the data to be converted.

number a pointer to an unsigned 16-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 318 of file numbers.c.

References st_data_get(), st_length_get(), and uint16_conv_bl().

Referenced by cache_set_value(), config_value_set(), relay_set_value(), servers_set_value(), and xml_get_xpath_uint16().

5.58.1.29 size_t uint16_digits (uint16_t *number*)

Count the number of digits needed to represent a base-10 16-bit unsigned integer.

Returns:

the number of digits needed to represent the specified integer.

Definition at line 44 of file digits.c.

References length.

Referenced by serialize_uint16().

5.58.1.30 uint16_t uint16_get_no (stringer_t * *s*)

Fetch a 2 byte network order value from a stringer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 2 network order bytes in the buffer in host byte order. 0 on failure.

Definition at line 852 of file numbers.c.

References data, log_pedantic, st_data_get(), st_empty, and st_length_get().

5.58.1.31 stringer_t* uint16_put_no (uint16_t *val*)

Create a stringer, storing a 2-byte value in network byte order.

Parameters:

s a pointer to the data buffer where the value will be inserted.

val the 2 byte value to be placed in the specified buffer, in network byte order.

Returns:

Pointer to stringer on success, NULL on failure.

Definition at line 774 of file numbers.c.

References data, log_pedantic, st_alloc(), st_data_get(), and st_length_set().

5.58.1.32 uint32_t uint24_get_no (stringer_t * s)

Fetch a 3 byte network order value from a stringer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 3 network order bytes in the buffer in host byte order. 0 on failure.

Definition at line 827 of file numbers.c.

References data, log_pedantic, st_data_get(), st_empty, and st_length_get().

5.58.1.33 stringer_t* uint24_put_no (uint32_t val)

Create a stringer, storing a 3-byte value in network byte order.

Parameters:

s a pointer to the data buffer where the value will be inserted.

val the 3 byte value to be placed in the specified buffer, in network byte order.

Returns:

Pointer to stringer on success, NULL on failure.

Definition at line 745 of file numbers.c.

References data, log_pedantic, st_alloc(), st_data_get(), and st_length_set().

5.58.1.34 uint32_t uint32_clamp (uint32_t min, uint32_t max, uint32_t number)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp

max maximum value for the clamp

number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 62 of file clamp.c.

References log_pedantic.

5.58.1.35 bool_t uint32_conv_bl (void * *block*, size_t *length*, uint32_t * *number*)

Convert a numerical string into an unsigned 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to an unsigned 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 188 of file numbers.c.

References data, and log_pedantic.

Referenced by cache_config(), deserialize_uint32(), relay_config(), servers_config(), uint32_conv_ns(), and uint32_conv_st().

5.58.1.36 bool_t uint32_conv_ns (char * *string*, uint32_t * *number*)

Convert a null-terminated string into an unsigned 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to an unsigned 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 236 of file numbers.c.

References ns_length_get(), and uint32_conv_bl().

5.58.1.37 bool_t uint32_conv_st (stringer_t * *string*, uint32_t * *number*)

Convert a managed string into an unsigned 32-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to an unsigned 32-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 247 of file numbers.c.

References `st_data_get()`, `st_length_get()`, and `uint32_conv_bl()`.

Referenced by `cache_set_value()`, `config_value_set()`, `imap_fetch_body_part()`, `imap_search_messages_date_compare()`, `relay_set_value()`, `servers_set_value()`, and `xml_get_xpath_uint32()`.

5.58.1.38 `size_t uint32_digits (uint32_t number)`

Count the number of digits needed to represent a base-10 32-bit unsigned integer.

Returns:

the number of digits needed to represent the specified integer.

Definition at line 29 of file digits.c.

References `length`.

Referenced by `serialize_uint32()`.

5.58.1.39 `uint32_t uint32_get_no (stringer_t * s)`

Fetch a 4 byte network order value from a stringer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 4 network order bytes in the buffer in host byte order. 0 on failure.

Definition at line 801 of file numbers.c.

References `data`, `log_pedantic`, `st_data_get()`, `st_empty`, and `st_length_get()`.

5.58.1.40 `stringer_t* uint32_put_no (uint32_t val)`

Create a stringer, storing a 4-byte value in network byte order.

Parameters:

s a pointer to the data buffer where the value will be inserted.

val the 4 byte value to be placed in the specified buffer, in network byte order.

Returns:

Pointer to stringer on success, NULL on failure.

Definition at line 715 of file numbers.c.

References `data`, `log_pedantic`, `st_alloc()`, `st_data_get()`, and `st_length_set()`.

5.58.1.41 `uint64_t uint64_clamp (uint64_t min, uint64_t max, uint64_t number)`

Ensure a number is between the min and max values, otherwise return tuint_the boundary value.

Parameters:

min minimum value for the clamp

max maximum value for the clamp

number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 83 of file clamp.c.

References log_pedantic.

Referenced by stacie_derive_rounds(), and stmt_exec_affected_conn().

5.58.1.42 **bool_t uint64_conv_bl** (void * *block*, size_t *length*, uint64_t * *number*)

Convert a numerical string into an unsigned 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the numerical string to be converted.

length the length, in bytes, of the numerical data.

number a pointer to an unsigned 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 106 of file numbers.c.

References data, and log_pedantic.

Referenced by deserialize_uint64(), imap_parse_literal(), pop_num_parse(), pop_top_parse(), size_conv_bl(), smtp_check_receive_quota(), uint64_conv_ns(), uint64_conv_pl(), and uint64_conv_st().

5.58.1.43 **bool_t uint64_conv_ns** (char * *string*, uint64_t * *number*)

Convert a null-terminated string into an unsigned 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the null-terminated string to be converted.

number a pointer to an unsigned 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 154 of file numbers.c.

References ns_length_get(), and uint64_conv_bl().

Referenced by json_api_dispatch(), lib_load_xml(), and portal_endpoint().

5.58.1.44 bool_t uint64_conv_pl (placer_t *string*, uint64_t * *number*)

Convert a placer into an unsigned 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the placer to be converted.

number a pointer to an unsigned 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 176 of file numbers.c.

References pl_data_get(), pl_length_get(), and uint64_conv_bl().

Referenced by build_version_major(), build_version_minor(), build_version_patch(), smtp_parse_mail_from_path(), and stamp_counter_check().

5.58.1.45 bool_t uint64_conv_st (stringer_t * *string*, uint64_t * *number*)

Convert a managed string into an unsigned 64-bit integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to the managed string to be converted.

number a pointer to an unsigned 64-bit number where the result of the conversion will be stored.

Returns:

true on success or false on failure.

Definition at line 165 of file numbers.c.

References st_data_get(), st_length_get(), and uint64_conv_bl().

Referenced by cache_set_value(), config_value_set(), imap_narrow_messages(), imap_search_messages_range(), imap_search_messages_size(), portal_get_upload_attachment(), relay_set_value(), servers_set_value(), teacher_process(), and xml_get_xpath_uint64().

5.58.1.46 size_t uint64_digits (uint64_t *number*)

Count the number of digits needed to represent a base-10 64-bit unsigned integer.

Returns:

the number of digits needed to represent specified integer.

Definition at line 14 of file digits.c.

References length.

Referenced by mail_build_signature(), serialize_uint64(), sess_key(), and smtp_store_spamsig().

5.58.1.47 uint8_t uint8_clamp (uint8_t *min*, uint8_t *max*, uint8_t *number*)

Ensure a number is between the min and max values, otherwise return the boundary value.

Parameters:

min minimum value for the clamp
max maximum value for the clamp
number number to be clamped

Returns:

returns the min value if number is less than min, and returns max if number was greater than the max value, otherwise the number value is returned without any modifications. The original value for number is also returned if min is greater than max or max is less than min.

Definition at line 20 of file clamp.c.

References log_pedantic.

5.58.1.48 bool_t uint8_conv_bl (void * *block*, size_t *length*, uint8_t * *number*)

Convert a numerical string to an unsigned 8-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

block a pointer to the block of memory to be converted.
length the length, in bytes, of the numerical string.
number a pointer to an unsigned 8-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 330 of file numbers.c.

References data, and log_pedantic.

Referenced by ip_subnet_st(), uint8_conv_ns(), and uint8_conv_st().

5.58.1.49 bool_t uint8_conv_ns (char * *string*, uint8_t * *number*)

Convert a numerical string to an 8-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to a null-terminated string containing the data to be converted.
number a pointer to an unsigned 8-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 378 of file numbers.c.

References ns_length_get(), and uint8_conv_bl().

5.58.1.50 `bool_t uint8_conv_st (stringer_t * string, uint8_t * number)`

Convert a numerical string to an 8-bit unsigned integer.

Note:

This function checks for both numeric underflows and overflows.

Parameters:

string a pointer to a managed string containing the data to be converted.

number a pointer to an unsigned 8-bit integer that will hold the result of the conversion.

Returns:

true on success, or false on failure.

Definition at line 389 of file numbers.c.

References `st_data_get()`, `st_length_get()`, and `uint8_conv_bl()`.

Referenced by `cache_set_value()`, `config_value_set()`, `relay_set_value()`, `servers_set_value()`, and `xml_get_xpath_uint8()`.

5.58.1.51 `size_t uint8_digits (uint8_t number)`

Count the number of digits needed to represent a base-10 8-bit unsigned integer.

Returns:

the number of digits needed to represent the specified integer.

Definition at line 59 of file digits.c.

References `length`.

5.59 src/core/parsers/parsers.h File Reference

```
#include "numbers/numbers.h"
#include "formats/formats.h"
#include "special/special.h"
```

Data Structures

- struct [tok_state_t](#)

Functions

- [uint64_t time_datestamp](#) (void)
time.c
- [stringer_t * time_print_gmt](#) ([stringer_t](#) *s, [chr_t](#) *format, [time_t](#) moment)
Get a specified time as a formatted string converted to GMT.
- [stringer_t * time_print_local](#) ([stringer_t](#) *s, [chr_t](#) *format, [time_t](#) moment)
Get a specified time as a formatted string.
- [uint64_t time_till_midnight](#) (void)
Get the number of seconds until midnight.
- [uchr_t lower_chr](#) ([uchr_t](#) c)
Return the lowercase representation of a character.
- [stringer_t * lower_st](#) ([stringer_t](#) *s)
Transform a managed string (in-place) into lowercase.
- [uchr_t upper_chr](#) ([uchr_t](#) c)
Return the uppercase representation of a character.
- [stringer_t * upper_st](#) ([stringer_t](#) *s)
Transform a managed string (in-place) into uppercase.
- [uint64_t tok_get_count_st](#) ([stringer_t](#) *string, char token)
Count the number of times a token is found in a managed string.
- [uint64_t tok_get_count_bl](#) (void *block, [size_t](#) length, char token)
Count the number of times a token is found in a specified block of memory.
- [uint64_t str_tok_get_count_bl](#) (void *block, [size_t](#) length, [chr_t](#) *token, [size_t](#) toklen)
Count the number of times a string token is found in a specified block of memory.
- [int tok_get_pl](#) ([placer_t](#) string, char token, [uint64_t](#) fragment, [placer_t](#) *value)
Retrieve a specified token from a placer.
- [int tok_get_st](#) ([stringer_t](#) *string, char token, [uint64_t](#) fragment, [placer_t](#) *value)
Retrieve a specified token from a managed string.

- `int tok_get_bl` (void *block, size_t length, char token, uint64_t fragment, `placer_t` *value)
Retrieve a specified token from a block of data.
- `int tok_get_ns` (char *string, size_t length, char token, uint64_t fragment, `placer_t` *value)
Retrieve a specified token from a null-terminated string.
- `int str_tok_get_bl` (char *block, size_t length, `chr_t` *token, size_t toklen, uint64_t fragment, `placer_t` *value)
Retrieve a specified string-split token from a null-terminated string.
- `int tok_pop` (`tok_state_t` *state, `placer_t` *value)
- `void tok_pop_init_st` (`tok_state_t` *state, `stringer_t` *string, char token)
- `void tok_pop_init_bl` (`tok_state_t` *state, void *block, size_t length, char token)
- `bool_t pl_skip_characters` (`placer_t` *place, char *skipchars, size_t nchars)
Skip past any of the specified characters found at the beginning of the placer, and update the placer accordingly.
- `bool_t pl_skip_to_characters` (`placer_t` *place, char *skiptochars, size_t nchars)
Skip to the first instance of any of the specified characters in the placer, and update the placer accordingly.
- `bool_t pl_shrink_before_characters` (`placer_t` *place, char *shrinkchars, size_t nchars)
Truncate a placer to start before any of the specified characters, and update the placer accordingly.
- `bool_t pl_get_embraced` (`placer_t` str, `placer_t` *out, unsigned char opening, unsigned char closing, `bool_t` required)
Get a placer pointing to the text contained within a specified pair of opening and closing braces.
- `bool_t pl_update_start` (`placer_t` *place, size_t nchars, `bool_t` more)
Ensure that a placer contains at least a certain amount of data, and update it accordingly.
- `placer_t line_pl_ns` (char *string, uint64_t number)
Get a placer pointing to the specified line (' delimited) of a null-terminated string.
- `placer_t line_pl_pl` (`placer_t` string, uint64_t number)
Get a placer pointing to the specified line (' delimited) of another placer.
- `placer_t line_pl_st` (`stringer_t` *string, uint64_t number)
Get a placer pointing to the specified line (' delimited) of a managed string.
- `placer_t line_pl_bl` (char *block, size_t length, uint64_t number)
Get a placer pointing to the specified line (' delimited) of content in a data buffer.
- `placer_t pl_trim` (`placer_t` place)
trim.c
- `placer_t pl_trim_end` (`placer_t` place)
- `placer_t pl_trim_start` (`placer_t` place)
Trim the leading whitespace from a placer.
- `void st_trim` (`stringer_t` *string)
- `stringer_t * st_xor` (`stringer_t` *a, `stringer_t` *b, `stringer_t` *outcome)
bitwise.c

- [stringer_t * st_and](#) ([stringer_t *a](#), [stringer_t *b](#), [stringer_t *outcome](#))

Perform bitwise AND operation between two input strings.

- [stringer_t * st_or](#) ([stringer_t *a](#), [stringer_t *b](#), [stringer_t *outcome](#))

Perform bitwise OR operation between two input strings.

- [stringer_t * st_not](#) ([stringer_t *s](#), [stringer_t *outcome](#))

Perform bitwise NOT operation on a managed string.

5.59.1 Function Documentation

5.59.1.1 [placer_t line_pl_bl](#) ([char * block](#), [size_t length](#), [uint64_t number](#))

Get a placer pointing to the specified line (‘
’ delimited) of content in a data buffer.

Parameters:

- block*** a pointer to the block of data to be scanned.
- length*** the length, in bytes, of the specified data buffer.
- number*** the zero-based index of the line to be retrieved from the buffer.

Returns:

a null placer on failure, or a placer pointing to the specified line on success.

Definition at line 17 of file line.c.

References [log_pedantic](#), [mm_empty\(\)](#), [pl_init\(\)](#), and [pl_null\(\)](#).

Referenced by [line_pl_ns\(\)](#), [line_pl_pl\(\)](#), and [line_pl_st\(\)](#).

5.59.1.2 [placer_t line_pl_ns](#) ([char * string](#), [uint64_t number](#))

Get a placer pointing to the specified line (‘
’ delimited) of a null-terminated string.

See also:

[line_pl_bl\(\)](#)

Parameters:

- string*** a pointer to a null-terminated string to be scanned.
- number*** the zero-based index of the line to be retrieved from the string.

Returns:

a null placer on failure, or a placer pointing to the specified line on success.

Definition at line 60 of file line.c.

References [line_pl_bl\(\)](#), and [ns_length_get\(\)](#).

5.59.1.3 `placer_t line_pl_pl (placer_t string, uint64_t number)`

Get a placer pointing to the specified line (‘
’ delimited) of another placer.

Parameters:

string a pointer to the placer to be scanned.
number the zero-based index of the line to be retrieved from the placer.

Returns:

a null placer on failure, or a placer pointing to the specified line on success.

Definition at line 82 of file line.c.

References `line_pl_bl()`, `pl_char_get()`, and `pl_length_get()`.

5.59.1.4 `placer_t line_pl_st (stringer_t * string, uint64_t number)`

Get a placer pointing to the specified line (‘
’ delimited) of a managed string.

Parameters:

string a pointer to the managed string to be scanned.
number the zero-based index of the line to be retrieved from the managed string.

Returns:

a null placer on failure, or a placer pointing to the specified line on success.

Definition at line 71 of file line.c.

References `line_pl_bl()`, `st_char_get()`, and `st_length_get()`.

Referenced by `client_read_line()`, `con_read_line()`, and `imap_parse_literal()`.

5.59.1.5 `uchr_t lower_chr (uchr_t c)`

Return the lowercase representation of a character.

Parameters:

c the character to be transformed.

Returns:

the input character, in lowercase.

Definition at line 25 of file case.c.

Referenced by `auth_sanitize_address()`, `hex_decode_chr()`, `mm_cmp_ci_eq()`, `pop_user_parse()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, `smtp_parse_rcpt_to()`, `st_cmp_ci_ends()`, `st_cmp_ci_eq()`, `st_cmp_ci_starts()`, and `st_search_ci()`.

5.59.1.6 stringer_t* lower_st (stringer_t * s)

Transform a managed string (in-place) into lowercase.

Parameters:

s the managed string to be modified.

Returns:

NULL on error, or a pointer to the input managed string, in lowercase.

Definition at line 58 of file case.c.

References log_pedantic, and st_empty_out().

Referenced by contact_business_valid_email(), http_parse_header(), register_business_step1(), and smtp_rcpt_to().

5.59.1.7 bool_t pl_get_embraced (placer_t str, placer_t * out, unsigned char opening, unsigned char closing, bool_t required)

Get a placer pointing to the text contained within a specified pair of opening and closing braces.

Parameters:

str a placer pointing to the string to be parsed.

out a pointer to a placer that will store the string between the braces on success.

opening the character to be the opening brace of the sequence.

closing the character to be the closing brace of the sequence.

required if true, fail if no data was found between the pair of braces.

Returns:

true if the brace sequence was complete, or false if either component could not be located.

Definition at line 442 of file token.c.

References pl_char_get(), and pl_empty().

5.59.1.8 bool_t pl_shrink_before_characters (placer_t * place, char * shrinkchars, size_t nchars)

Truncate a placer to start before any of the specified characters, and update the placer accordingly.

Parameters:

place a pointer to a placer that will be updated to be truncated before any of the specified characters.

shrinkchars a pointer to a buffer containing bytes that will be skipped when they are found at the end of the placer.

nchars the number of characters to be tested in the collection in shrinkchars.

Returns:

true if the shrink operation completed before the end of the placer was reached, or false otherwise.

Definition at line 405 of file token.c.

References pl_char_get(), pl_empty(), and pl_length_get().

Referenced by portal_upload().

5.59.1.9 `bool_t pl_skip_characters (placer_t * place, char * skipchars, size_t nchars)`

Skip past any of the specified characters found at the beginning of the placer, and update the placer accordingly.

Parameters:

place a pointer to a placer that will be updated to skip past any of the specified characters.

skipchars a pointer to a buffer containing bytes that will be skipped at the beginning of the placer.

nchars the number of characters to be tested in the collection in skipchars.

Returns:

true if the skip operation completed before the end of the placer was reached, or false otherwise.

Definition at line 336 of file token.c.

References `pl_char_get()`, and `pl_empty()`.

Referenced by `portal_upload()`.

5.59.1.10 `bool_t pl_skip_to_characters (placer_t * place, char * skiptochars, size_t nchars)`

Skip to the first instance of any of the specified characters in the placer, and update the placer accordingly.

Parameters:

place a pointer to a placer that will be updated to skip to any of the specified characters.

skiptochars a pointer to a buffer containing bytes that will be skipped to when they are first found in the placer.

nchars the number of characters to be tested in the collection in skiptochars.

Returns:

true if the skip operation completed before the end of the placer was reached, or false otherwise.

Definition at line 372 of file token.c.

References `pl_char_get()`, and `pl_empty()`.

Referenced by `portal_upload()`.

5.59.1.11 `placer_t pl_trim (placer_t place)`

[trim.c](#)

Definition at line 42 of file trim.c.

References `pl_char_get()`, `pl_init()`, `pl_length_get()`, and `pl_null()`.

Referenced by `nvp_parse()`, `prime_pem_unwrap()`, and `smtp_parse_mail_from_path()`.

5.59.1.12 `placer_t pl_trim_end (placer_t place)`

Definition at line 84 of file trim.c.

References `pl_char_get()`, `pl_init()`, `pl_length_get()`, and `pl_null()`.

Referenced by `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, and `smtp_parse_rcpt_to()`.

5.59.1.13 `placer_t pl_trim_start (placer_t place)`

Trim the leading whitespace from a placer.

Parameters:

place a placer containing the string to have its leading whitespace trimmed.

Returns:

a placer pointing to the trimmed value inside the originally specified input string.

Definition at line 67 of file trim.c.

References `pl_char_get()`, `pl_init()`, `pl_length_get()`, and `pl_null()`.

Referenced by `get_header_opt()`.

5.59.1.14 `bool_t pl_update_start (placer_t * place, size_t nchars, bool_t more)`

Ensure that a placer contains at least a certain amount of data, and update it accordingly.

Parameters:

place a pointer to the placer containing the original data, which will be updated on success.

nchars the minimum number of characters that the placer needs to contain to pass the test.

more if true, more characters following the minimum buffer size are necessary.

Returns:

true if the placer meets the minimum size check, or false if it does not.

Definition at line 485 of file token.c.

5.59.1.15 `stringer_t* st_and (stringer_t * a, stringer_t * b, stringer_t * output)`

Perform bitwise AND operation between two input strings.

Parameters:

a First stringer input.

b Second stringer input.

output Stringer in which the result is stored, if no stringer is provided (`output = NULL`) then a new stringer will be allocated for the result.

Returns:

Pointer to output if a valid output stringer was provided, Pointer to a newly allocated stringer if no stringer was specified for the output, NULL on error.

Definition at line 149 of file bitwise.c.

References `bitwise_and()`, and `st_bitwise()`.

5.59.1.16 `stringer_t* st_not (stringer_t * s, stringer_t * output)`

Perform bitwise NOT operation on a managed string.

Parameters:

s Stringer input.

output Stringer in which the result is stored, if no stringer is provided (output = NULL) then a new stringer will be allocated for the result.

Returns:

Pointer to output if a valid output stringer was provided, or a pointer to a newly allocated stringer if no stringer was specified for the output, NULL on error.

Definition at line 75 of file bitwise.c.

References data, log_pedantic, st_alloc(), st_avail_get(), st_data_get(), st_empty, st_free(), st_length_get(), st_length_set(), st_valid_avail(), and st_valid_destination().

5.59.1.17 stringer_t* st_or (stringer_t * *a*, stringer_t * *b*, stringer_t * *output*)

Perform bitwise OR operation between two input strings.

Parameters:

a First stringer input.

b Second stringer input.

output Stringer in which the result is stored, if no stringer is provided (output = NULL) then a new stringer will be allocated for the result.

Returns:

Pointer to output if a valid output stringer was provided, Pointer to a newly allocated stringer if no stringer was specified for the output, NULL on error.

Definition at line 123 of file bitwise.c.

References bitwise_or(), and st_bitwise().

5.59.1.18 void st_trim (stringer_t * *string*)

Definition at line 15 of file trim.c.

References mm_move(), mm_wipe(), st_avail_get(), st_char_get(), st_length_get(), and st_length_set().

Referenced by mail_header_fetch_cleaned(), and process_find_pid().

5.59.1.19 stringer_t* st_xor (stringer_t * *a*, stringer_t * *b*, stringer_t * *output*)

bitwise.c bitwise.c

Parameters:

a First stringer input.

b Second stringer input.

output Stringer in which the result is stored, if no stringer is provided (output = NULL) then a new stringer will be allocated for the result.

Returns:

Pointer to output if a valid output stringer was provided, or a pointer to a newly allocated stringer if no stringer was specified for the output, NULL on error.

Definition at line 136 of file bitwise.c.

References `bitwise_xor()`, and `st_bitwise()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `signature_full_get()`, `signature_full_verify()`, `signature_tree_get()`, `signature_tree_verify()`, `slots_key()`, `slots_set()`, `stacie_decrypt()`, `stacie_encrypt()`, and `stacie_realm_key()`.

5.59.1.20 `int str_tok_get_bl(char *block, size_t length, chr_t *token, size_t toklen, uint64_t fragment, placer_t *value)`

Retrieve a specified string-split token from a null-terminated string.

Parameters:

block a pointer to the block of memory to be tokenized.

length the maximum number of characters to be scanned from the input data.

token the token string that will be used to split the data.

toklen the length, in bytes, of the token string.

fragment the zero-indexed token number to be extracted from the data.

value a pointer to a placer that will receive the value of the extracted token on success, or `pl_null()` on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one in the string.

Definition at line 290 of file token.c.

References `mm_empty()`, `pl_char_get()`, `pl_init()`, `pl_length_get()`, `pl_null()`, `placer_t`, and `st_search_cs()`.

Referenced by `portal_upload()`.

5.59.1.21 `uint64_t str_tok_get_count_bl(void *block, size_t length, chr_t *token, size_t toklen)`

Count the number of times a string token is found in a specified block of memory.

Parameters:

block a pointer to a block of memory to be scanned.

length the length, in bytes, of the block of memory to be scanned.

token a pointer to the string token being used to split the input data.

toklen the length, in bytes, of the specified token.

Returns:

the number of times the string token was found in the block of memory, or 0 on failure.

Definition at line 257 of file token.c.

References `count`, `mm_empty()`, `pl_init()`, `pl_length_get()`, `placer_t`, and `st_search_cs()`.

Referenced by `portal_upload()`.

5.59.1.22 `uint64_t time_datestamp(void)`

[time.c](#) [time.c](#)

Returns:

0 on failure or a 64-bit unsigned integer containing the formatted current date on success.

Definition at line 36 of file time.c.

Referenced by api_endpoint_auth(), imap_login(), log_rotate(), log_start(), pattern_update(), pop_pass(), portal_endpoint_auth(), process_maint(), smtp_auth_login(), and smtp_auth_plain().

5.59.1.23 stringer_t* time_print_gmt (stringer_t * s, chr_t * format, time_t moment)

Get a specified time as a formatted string converted to GMT.

Parameters:

- s* a managed string where the formatted time is to be stored.
- format* a null-terminated string containing the format of the time string.
- moment* the specified time to be formatted, as a UNIX time value.

Returns:

NULL on failure, or a pointer to the managed string containing the result on success.

Definition at line 92 of file time.c.

References log_pedantic, mm_wipe(), st_avail_get(), st_char_get(), and st_length_set().

Referenced by http_response_cookie(), http_response_header(), and http_response_options().

5.59.1.24 stringer_t* time_print_local (stringer_t * s, chr_t * format, time_t moment)

Get a specified time as a formatted string.

Parameters:

- s* a managed string where the formatted time is to be stored.
- format* a null-terminated string containing the format of the time string.
- moment* the specified time to be formatted, as a UNIX time value.

Returns:

NULL on failure, or a pointer to the managed string containing the result on success.

Definition at line 61 of file time.c.

References log_pedantic, mm_wipe(), st_avail_get(), st_char_get(), and st_length_set().

Referenced by imap_id(), and register_data_insert_user().

5.59.1.25 uint64_t time_till_midnight (void)

Get the number of seconds until midnight.

Returns:

an unsigned 64-bit integer containing the number of seconds until midnight, or 86400 on failure.

Definition at line 14 of file time.c.

Referenced by process_maint().

5.59.1.26 `int tok_get_bl (void * block, size_t length, char token, uint64_t fragment, placer_t * value)`

Retrieve a specified token from a block of data.

See also:

[tok_get_ns\(\)](#)

Parameters:

block a pointer to the memory block to be tokenized.

length the maximum number of characters to be scanned at the input data block.

token the token character that will be used to split the data block.

fragment the zero-indexed token number to be extracted from the data block.

value a pointer to a placer that will receive the value of the extracted token on success, or [pl_null\(\)](#) on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one found in the data block.

Definition at line 128 of file token.c.

References [tok_get_ns\(\)](#).

Referenced by [nvp_parse\(\)](#), [smtp_parse_mail_from_path\(\)](#), [tok_get_pl\(\)](#), and [tok_get_st\(\)](#).

5.59.1.27 `uint64_t tok_get_count_bl (void * block, size_t length, char token)`

Count the number of times a token is found in a specified block of memory.

Parameters:

block a pointer to a block of memory to be scanned.

length the length, in bytes, of the block of memory to be scanned.

token a character specifying the token to be searched.

Returns:

the number of times the token was found in the block of memory, or 0 on failure.

Definition at line 17 of file token.c.

References [count](#), [log_pedantic](#), and [mm_empty\(\)](#).

Referenced by [smtp_parse_mail_from_path\(\)](#), and [tok_get_count_st\(\)](#).

5.59.1.28 `uint64_t tok_get_count_st (stringer_t * string, char token)`

Count the number of times a token is found in a managed string.

See also:

[tok_get_count_bl\(\)](#)

Parameters:

string a pointer to the managed string to be scanned.

token a character specifying the token to be searched.

Returns:

the number of times the token was found in the managed string, or 0 on failure.

Definition at line 49 of file token.c.

References `st_data_get()`, `st_length_get()`, and `tok_get_count_bl()`.

Referenced by `get_header_opt()`, `get_header_value_noopt()`, `http_parse_method()`, `http_parse_pairs()`, `imap_fetch_body_part()`, `imap_narrow_messages()`, `imap_search_messages_date()`, `imap_search_messages_date_compare()`, `imap_search_messages_range()`, `ip_subnet_st()`, `mail_domain_get()`, `mail_mime_type_parameters()`, `portal_get_upload_attachment()`, and `stamp_counter_check()`.

5.59.1.29 int tok_get_ns (char * string, size_t length, char token, uint64_t fragment, placer_t * value)

Retrieve a specified token from a null-terminated string.

Parameters:

string a pointer to the null-terminated string to be tokenized.

length the maximum number of characters to be scanned from the input string.

token the token character that will be used to split the string.

fragment the zero-indexed token number to be extracted from the string.

value a pointer to a placer that will receive the value of the extracted token on success, or `pl_null()` on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one in the string.

Definition at line 63 of file token.c.

References `log_pedantic`, `mm_empty()`, `pl_init()`, and `pl_null()`.

Referenced by `ip_subnet_st()`, `lib_load_openssl()`, and `tok_get_bl()`.

5.59.1.30 int tok_get_pl (placer_t string, char token, uint64_t fragment, placer_t * value)

Retrieve a specified token from a placer.

See also:

`token_get_ns()`

Parameters:

string a pointer to the placer to be tokenized.

token the token character that will be used to split the placer.

fragment the zero-indexed token number to be extracted from the placer.

value a pointer to a placer that will receive the value of the extracted token on success, or `pl_null()` on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one found in the placer.

Definition at line 156 of file token.c.

References `pl_data_get()`, `pl_length_get()`, and `tok_get_bl()`.

Referenced by `http_data_value_parse()`, `http_parse_context()`, `http_parse_method()`, and `smtp_parse_mail_from_path()`.

5.59.1.31 `int tok_get_st (stringer_t * string, char token, uint64_t fragment, placer_t * value)`

Retrieve a specified token from a managed string.

See also:

[tok_get_ns\(\)](#)

Parameters:

string a pointer to the managed string to be tokenized.

token the token character that will be used to split the managed string.

fragment the zero-indexed token number to be extracted from the managed string.

value a pointer to a placer that will receive the value of the extracted token on success, or [pl_null\(\)](#) on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one found in the managed string.

Definition at line 142 of file token.c.

References [st_data_get\(\)](#), [st_length_get\(\)](#), and [tok_get_bl\(\)](#).

Referenced by [build_version_major\(\)](#), [build_version_minor\(\)](#), [build_version_patch\(\)](#), [get_header_opt\(\)](#), [get_header_value_noopt\(\)](#), [http_parse_context\(\)](#), [http_parse_pairs\(\)](#), [imap_fetch_body_part\(\)](#), [imap_folder_create\(\)](#), [imap_folder_remove\(\)](#), [imap_folder_rename\(\)](#), [imap_narrow_messages\(\)](#), [imap_parse_address_part\(\)](#), [imap_search_messages_date\(\)](#), [imap_search_messages_date_compare\(\)](#), [imap_search_messages_range\(\)](#), [mail_domain_get\(\)](#), [mail_mime_type_parameters\(\)](#), [portal_get_upload_attachment\(\)](#), [smtp_auth_plain\(\)](#), and [stamp_counter_check\(\)](#).

5.59.1.32 `int tok_pop (tok_state_t * state, placer_t * value)`

on error the function returns -1 and sets value to EMPTY_PLACER on success the function returns 0 and stores the token in the placer pointed at by value if the end is hit, then the function returns 1, and places any remaining data in value

Definition at line 188 of file token.c.

References [pl_init\(\)](#), [pl_null\(\)](#), [tok_state_t::position](#), [tok_state_t::remaining](#), and [tok_state_t::token](#).

Referenced by [nvp_parse\(\)](#), and [prime_pem_unwrap\(\)](#).

5.59.1.33 `void tok_pop_init_bl (tok_state_t * state, void * block, size_t length, char token)`

Definition at line 161 of file token.c.

References [tok_state_t::block](#), [tok_state_t::length](#), [tok_state_t::position](#), [tok_state_t::remaining](#), and [tok_state_t::token](#).

5.59.1.34 `void tok_pop_init_st (tok_state_t * state, stringer_t * string, char token)`

Definition at line 172 of file token.c.

References [tok_state_t::block](#), [tok_state_t::length](#), [tok_state_t::position](#), [tok_state_t::remaining](#), [st_char_get\(\)](#), [st_length_get\(\)](#), and [tok_state_t::token](#).

Referenced by [nvp_parse\(\)](#), and [prime_pem_unwrap\(\)](#).

5.59.1.35 `uchar_t upper_chr (uchar_t c)`

Return the uppercase representation of a character.

Parameters:

c the character to be transformed.

Returns:

the input character, in uppercase.

Definition at line 15 of file case.c.

5.59.1.36 stringer_t* upper_st (stringer_t * s)

Transform a managed string (in-place) into uppercase.

Parameters:

s the managed string to be modified.

Returns:

NULL on error, or a pointer to the input managed string, in uppercase.

Definition at line 35 of file case.c.

References `log_pedantic`, and `st_empty_out()`.

Referenced by `http_response_allow_cross()`, `imap_fetch_body_tag()`, `imap_fetch_bodystructure()`, and `mail_mime_type_parameters()`.

5.60 src/providers/parsers/parsers.h File Reference

Functions

- [bool_t lib_load_jansson](#) (void)
json.c
- [chr_t * lib_version_jansson](#) (void)
Return the version string for libjansson.
- [bool_t lib_load_xml](#) (void)
xml.c
- [bool_t xml_start](#) (void)
Initialize the xml parser library.
- [chr_t * xml_get_xpath_ns](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the content of an evaluated xpath expression as a null-terminated string.
- [const chr_t * lib_version_xml](#) (void)
Return the version string of libxml.
- [int16_t xml_get_xpath_int16](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a signed 16-bit integer.
- [int32_t xml_get_xpath_int32](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a signed 32-bit integer.
- [int64_t xml_get_xpath_int64](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a signed 64-bit integer.
- [int8_t xml_get_xpath_int8](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a signed 8-bit integer.
- [int_t xml_xpath_set_namespace](#) (xmlXPathContextPtr ctxt, xmlChar *prefix, xmlChar *ns_uri)
Set the namespace for an xpath context.
- [size_t xml_get_xpath_node_count](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
This function is not currently being called by any other part of the code.
- [stringer_t * xml_dump_doc](#) (xmlDocPtr doc)
Get an xml document object as a string.
- [stringer_t * xml_get_xpath_st](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a managed string.
- [stringer_t * xml_node_get_content_st](#) (xmlNodePtr node)
Get the content of an xml node as a managed string.
- [uint16_t xml_get_xpath_uint16](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as an unsigned 16-bit integer.
- [uint32_t xml_get_xpath_uint32](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)

Get the value of a node element or attribute selected by an xpath expression as an unsigned 32-bit integer.

- `uint64_t xml_get_xpath_uint64` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as an unsigned 64-bit integer.
- `uint8_t xml_get_xpath_uint8` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as an unsigned 8-bit integer.
- `bool_t xml_set_xpath_uint64` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query, uint64_t val)
Set the value of a node element selected by an xpath expression, as a 64-bit unsigned integer.
- `bool_t xml_set_xpath_ns` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query, `uchr_t` *val)
Set the value of a node element selected by an xpath expression, as a null-terminated string.
- `bool_t xml_set_xpath_property` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query, `uchr_t` *name, `uchr_t` *val)
Set the value of a specified property of a node element selected by an xpath expression.
- `void xml_error` (void *ctx, const `chr_t` *format,...)
A function handler for displaying xml parser error messages to the user.
- `void xml_free_doc` (xmlDocPtr doc)
Free an xml document.
- `void xml_free_parser_ctx` (xmlParserCtxtPtr ctx)
Destroy an xml parser context.
- `void xml_free_xpath_ctx` (xmlXPathContextPtr ctx)
Free an xml xpath context.
- `void xml_free_xpath_obj` (xmlXPathObjectPtr obj)
Free an xml xpath object.
- `void xml_node_free` (xmlNodePtr node)
Free an xml node.
- `void xml_node_set_content` (xmlNodePtr node, `uchr_t` *content)
Set the content of an xml node.
- `void xml_stop` (void)
Cleanup the state and allocated memory of the xml parser in preparation to be shutdown.
- `xmlAttrPtr xml_node_set_property` (xmlNodePtr node, `uchr_t` *name, `uchr_t` *value)
Set an attribute of an xml node.
- `xmlChar * xml_encode` (xmlDocPtr doc, `stringer_t` *string)
Encode an xml string, performing proper replacement of defined entities and non-ASCII values.
- `xmlDocPtr xml_create_doc` (xmlParserCtxtPtr ctx, const `chr_t` *buffer, `int_t` size, const `chr_t` *url, const `chr_t` *encoding, `int_t` options)
Create a new xml document object from specified user data.
- `xmlNodePtr xml_node_add_sibling` (xmlNodePtr current, xmlNodePtr element)
Add an xml node to the list of siblings of another xml node.

- xmlNodePtr [xml_node_new](#) (uchar_t *name)
Create an xml node.
- xmlParserCtxtPtr [xml_create_parser_ctx](#) (void)
Create and initialize a new xml parser context.
- xmlXPathContextPtr [xml_create_xpath_ctx](#) (xmlDocPtr doc)
Create an xpath context for an xml document.
- xmlXPathObjectPtr [xml_xpath_eval](#) (const uchar_t *xpath, xmlXPathContextPtr ctx)
Evaluate an xml xpath expression.
- bool_t [lib_load_utf8proc](#) (void)
utf.c
- chr_t * [lib_version_utf8proc](#) (void)
Return the shared library version string for libutf8proc.
- bool_t [utf8_valid_st](#) (stringer_t *s)
Determine whether a managed string is a properly formatted UTF8 string.
- size_t [utf8_length_st](#) (stringer_t *s)
Determine how many valid Unicode characters the string has.
- const chr_t * [utf8_error_string](#) (ssize_t error_code)
Exchange an error code for a string explaining the nature of the UTF8 error.

5.60.1 Function Documentation

5.60.1.1 bool_t lib_load_jansson (void)

[json.c](#) [json.c](#)

Returns:

true on success or false on failure.

Definition at line 22 of file [json.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.60.1.2 bool_t lib_load_utf8proc (void)

[utf.c](#) [utf.c](#)

Returns:

true on success or false on failure.

Definition at line 23 of file [utf8.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.60.1.3 `bool_t lib_load_xml (void)`

[xml.c](#) [xml.c](#)

Returns:

true on success or false on failure.

Definition at line 26 of file `xml.c`.

References `lib_symbols()`, `M_BIND`, `symbol_t`, `uint64_conv_ns()`, `xml_version_string`, and `xmlParserVersion_d`.

Referenced by `lib_load()`.

5.60.1.4 `chr_t* lib_version_jansson (void)`

Return the version string for libjansson.

Returns:

a pointer to a character string containing the libjansson version information.

Definition at line 14 of file `json.c`.

References `jansson_version_d`.

Referenced by `lib_load()`.

5.60.1.5 `chr_t* lib_version_utf8proc (void)`

Return the shared library version string for libutf8proc.

Returns:

a pointer to a character string containing the libutf8proc version information.

Definition at line 15 of file `utf8.c`.

References `utf8proc_version_d`.

Referenced by `lib_load()`.

5.60.1.6 `const chr_t* lib_version_xml (void)`

Return the version string of libxml.

Returns:

a pointer to a character string containing the libxml version information.

Definition at line 17 of file `xml.c`.

References `xml_version_string`.

Referenced by `lib_load()`.

5.60.1.7 `const chr_t* utf8_error_string (ssize_t error_code)`

Exchange an error code for a string explaining the nature of the UTF8 error.

Parameters:

error_code the numeric error returned by the internal UTF8 mapping logic.

Returns:

A string constant with the error message. If the code goes unrecognized, then a generic message is returned.

Definition at line 42 of file utf8.c.

References utf8proc_errmsg_d.

Referenced by utf8_length_st().

5.60.1.8 size_t utf8_length_st (stringer_t * s)

Determine how many valid Unicode characters the string has.

Parameters:

s the managed string to be tested.

Returns:

The number of codepoints found if the string is valid, otherwise 0.

Definition at line 51 of file utf8.c.

References bytes, log_pedantic, st_empty_out(), utf8_error_string(), utf8proc_get_property_d, and utf8proc_iterate_d.

Referenced by auth_login(), register_business_validate_password(), register_data_insert_user(), and stacie_derive_rounds().

5.60.1.9 bool_t utf8_valid_st (stringer_t * s)

Determine whether a managed string is a properly formatted UTF8 string.

Parameters:

s the managed string to be tested.

Returns:

Valid strings return true, while invalid strings return false.

Definition at line 90 of file utf8.c.

References bytes, log_pedantic, st_empty_out(), and utf8proc_iterate_d.

5.60.1.10 xmlDocPtr xml_create_doc (xmlParserCtxtPtr ctx, const chr_t * buffer, int_t size, const chr_t * url, const chr_t * encoding, int_t options)

Create a new xml document object from specified user data.

See also:

xmlCtxtReadMemory()

Parameters:

ctx a pointer to the xml context to be used for the parsing operation.

buffer a null-terminated string containing the xml data to be parsed.

size the length, in bytes, of the xml data buffer to be parsed.

url a null-terminated string containing the base url to be used for the document.

encoding a null-terminated string specifying the document encoding type, or NULL for default.

options a value containing a mask of xml parser options of type xmlParserOption.

Returns:

NULL on failure, or a pointer to the xml document tree of the parsed xml text on success.

Definition at line 1051 of file xml.c.

References log_pedantic, and xmlCtxtReadMemory_d.

Referenced by http_page_get().

5.60.1.11 xmlParserCtxtPtr xml_create_parser_ctx (void)

Create and initialize a new xml parser context.

Returns:

NULL on failure, or a newly initialized xml parser context on success.

Definition at line 961 of file xml.c.

References log_pedantic, xml_error(), and xmlNewParserCtxt_d.

Referenced by http_page_get().

5.60.1.12 xmlXPathContextPtr xml_create_xpath_ctx (xmlDocPtr doc)

Create an xpath context for an xml document.

Parameters:

doc a pointer to the input xml document object.

Returns:

NULL on failure, or a pointer to the new xpath context on success.

Definition at line 917 of file xml.c.

References log_pedantic, and xmlXPathNewContext_d.

Referenced by http_page_get().

5.60.1.13 stringer_t* xml_dump_doc (xmlDocPtr doc)

Get an xml document object as a string.

Parameters:

doc a pointer to the xml document object to be serialized.

Returns:

NULL on failure or a pointer to a managed string containing the specified xml document's serialized data on success.

Definition at line 1018 of file xml.c.

References mm_free(), st_import(), and xmlDocDumpFormatMemory_d.

Referenced by contact_print_form(), contact_print_message(), portal_print_login(), statistics_process(), teacher_print_form(), and teacher_print_message().

5.60.1.14 xmlChar* xml_encode (xmlDocPtr *doc*, stringer_t * *string*)

Encode an xml string, performing proper replacement of defined entities and non-ASCII values.

Parameters:

doc a pointer to the xml document containing the DTD to be used for the encoding process.

string a managed string containing the xml data to be encoded.

NULL on failure, or a newly allocated null-terminated string containing the encoded xml data on success.

Definition at line 69 of file xml.c.

References st_data_get(), and xmlEncodeEntitiesReentrant_d.

Referenced by contact_print_form().

5.60.1.15 void xml_error (void * *ctx*, const chr_t * *format*, ...)

A function handler for displaying xml parser error messages to the user.

Parameters:

ctx a placeholder; ignored.

format a null-terminated string containing a format string for the error message to be displayed.

... a variable argument list containing the parameters to the format string.

Returns:

This function returns no value.

Definition at line 854 of file xml.c.

References log_internal(), M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE_DISABLE, and M_LOG_TIME_DISABLE.

Referenced by xml_create_parser_ctx().

5.60.1.16 void xml_free_doc (xmlDocPtr *doc*)

Free an xml document.

Parameters:

doc a pointer to the xml document to be freed.

Returns:

This function returns no value.

Definition at line 1001 of file xml.c.

References log_pedantic, and xmlFreeDoc_d.

Referenced by http_page_free().

5.60.1.17 void xml_free_parser_ctx (xmlParserCtxtPtr *ctx*)

Destroy an xml parser context.

Parameters:

ctx a pointer to the xml parser context to be freed.

Returns:

This function returns no value.

Definition at line 984 of file xml.c.

References log_pedantic, and xmlFreeParserCtxt_d.

Referenced by http_page_free().

5.60.1.18 void xml_free_xpath_ctx (xmlXPathContextPtr *ctx*)

Free an xml xpath context.

Parameters:

ctx a pointer to the xml xpath context to be freed.

Returns:

This function returns no value.

Definition at line 883 of file xml.c.

References log_pedantic, and xmlXPathFreeContext_d.

Referenced by http_page_free().

5.60.1.19 void xml_free_xpath_obj (xmlXPathObjectPtr *obj*)

Free an xml xpath object.

Parameters:

obj a pointer to the xml xpath object to be freed.

Returns:

This function returns no value.

Definition at line 900 of file xml.c.

References log_pedantic, and xmlXPathFreeObject_d.

Referenced by xml_set_xpath_ns(), xml_set_xpath_property(), and xml_set_xpath_uint64().

5.60.1.20 int16_t xml_get_xpath_int16 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as a signed 16-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a signed 16-bit integer.

Definition at line 503 of file xml.c.

References `int16_conv_st()`, `log_pedantic`, `ns_length_get()`, `PLACER`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.60.1.21 int32_t xml_get_xpath_int32 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as a signed 32-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a signed 32-bit integer.

Definition at line 448 of file xml.c.

References `int32_conv_st()`, `log_pedantic`, `ns_length_get()`, `PLACER`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.60.1.22 int64_t xml_get_xpath_int64 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as a signed 64-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a signed 64-bit integer.

Definition at line 393 of file xml.c.

References `int64_conv_st()`, `log_pedantic`, `ns_length_get()`, `PLACER`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.60.1.23 int8_t xml_get_xpath_int8 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as a signed 8-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a signed 8-bit integer.

Definition at line 558 of file xml.c.

References `int8_conv_st()`, `log_pedantic`, `ns_length_get()`, `PLACER`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.60.1.24 `size_t xml_get_xpath_node_count (xmlXPathContextPtr xpath_ctx, xmlChar * xpath_query)`

This function is not currently being called by any other part of the code.

Definition at line 817 of file xml.c.

References `log_pedantic`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.60.1.25 `chr_t* xml_get_xpath_ns (xmlXPathContextPtr xpath_ctx, xmlChar * xpath_query)`

Get the content of an evaluated xpath expression as a null-terminated string.

Parameters:

ctx a pointer to the xpath context to be used for the evaluation.

xpath_query a null-terminated string containing the xpath expression to be evaluated.

Returns:

NULL on failure, or a null-terminated string containing the value of the xpath expression's specified node on success.

Definition at line 766 of file xml.c.

References `log_pedantic`, `ns_dupe()`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.60.1.26 `stringer_t* xml_get_xpath_st (xmlXPathContextPtr xpath_ctx, xmlChar * xpath_query)`

Get the value of a node element or attribute selected by an xpath expression as a managed string.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a managed string.

Definition at line 613 of file xml.c.

References `log_pedantic`, `ns_length_get()`, `st_import()`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.60.1.27 `uint16_t xml_get_xpath_uint16 (xmlXPathContextPtr xpath_ctx, xmlChar * xpath_query)`

Get the value of a node element or attribute selected by an xpath expression as an unsigned 16-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as an unsigned 16-bit integer.

Definition at line 283 of file xml.c.

References `log_pedantic`, `ns_length_get()`, `PLACER`, `uint16_conv_st()`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.60.1.28 uint32_t xml_get_xpath_uint32 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as an unsigned 32-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as an unsigned 32-bit integer.

Definition at line 229 of file xml.c.

References log_pedantic, ns_length_get(), PLACER, uint32_conv_st(), xmlXPathEvalExpression_d, and xmlXPathFreeObject_d.

5.60.1.29 uint64_t xml_get_xpath_uint64 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as an unsigned 64-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as an unsigned 64-bit integer.

Definition at line 174 of file xml.c.

References log_pedantic, ns_length_get(), PLACER, uint64_conv_st(), xmlXPathEvalExpression_d, and xmlXPathFreeObject_d.

5.60.1.30 uint8_t xml_get_xpath_uint8 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as an unsigned 8-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as an unsigned 8-bit integer.

Definition at line 338 of file xml.c.

References log_pedantic, ns_length_get(), PLACER, uint8_conv_st(), xmlXPathEvalExpression_d, and xmlXPathFreeObject_d.

5.60.1.31 xmlNodePtr xml_node_add_sibling (xmlNodePtr *current*, xmlNodePtr *element*)

Add an xml node to the list of siblings of another xml node.

Note:

If the sibling node was already inserted into a document it will first be unlinked.

Parameters:

current a pointer to the xml node to which the sibling node will be added.

element a pointer to the sibling node to be added to the target node.

Returns:

NULL on failure, or a pointer to the sibling node on success.

Definition at line 94 of file xml.c.

References xmlAddSibling_d.

Referenced by contact_business_add_error(), and teacher_add_error().

5.60.1.32 void xml_node_free (xmlNodePtr node)

Free an xml node.

Parameters:

node a pointer to the xml node to be freed.

Returns:

This function returns no value.

Definition at line 57 of file xml.c.

References xmlFreeNode_d.

Referenced by contact_business_add_error(), and teacher_add_error().

5.60.1.33 stringer_t* xml_node_get_content_st (xmlNodePtr node)

Get the content of an xml node as a managed string.

Parameters:

node a pointer to the xml node to be queried.

Returns:

NULL on failure, or a pointer to a managed string

Definition at line 114 of file xml.c.

References log_pedantic, st_import(), xmlBufferContent_d, xmlBufferCreate_d, xmlBufferFree_d, xmlBufferLength_d, and xmlNodeBufGetContent_d.

5.60.1.34 xmlNodePtr xml_node_new (uchr_t * name)

Create an xml node.

Parameters:

name the name of the new xml node.

Returns:

NULL on failure, or a pointer to the newly allocated and initialized xml node on success.

Definition at line 104 of file xml.c.

References xmlNewNode_d.

Referenced by contact_business_add_error(), and teacher_add_error().

5.60.1.35 void xml_node_set_content (xmlNodePtr *node*, uchr_t * *content*)

Set the content of an xml node.

Parameters:

node a pointer to the xml node to be set.

content a null-terminated string containing the new content of the xml node.

Returns:

This function returns no value.

Definition at line 150 of file xml.c.

References xmlNodeSetContent_d.

Referenced by contact_business_add_error(), portal_print_login(), teacher_add_error(), xml_set_xpath_ns(), and xml_set_xpath_uint64().

5.60.1.36 xmlAttrPtr xml_node_set_property (xmlNodePtr *node*, uchr_t * *name*, uchr_t * *value*)

Set an attribute of an xml node.

Parameters:

node a pointer to the xml node to be set.

name a null-terminated string containing the name of the node attribute to be set.

value a null-terminated string containing the new value of the specified xml node's attribute.

Returns:

NULL on failure, or a pointer to node's attribute on success.

Definition at line 163 of file xml.c.

References xmlSetProp_d.

Referenced by contact_business_add_error(), teacher_add_error(), and xml_set_xpath_property().

5.60.1.37 bool_t xml_set_xpath_ns (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*, uchr_t * *val*)

Set the value of a node element selected by an xpath expression, as a null-terminated string.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

val the new value of the selected node, as a null-terminated string.

Returns:

true if the value was set successfully, or false on failure.

Definition at line 702 of file xml.c.

References log_pedantic, xml_free_xpath_obj(), xml_node_set_content(), and xml_xpath_eval().

Referenced by contact_print_form(), contact_print_message(), statistics_process(), teacher_print_form(), and teacher_print_message().

5.60.1.38 bool_t xml_set_xpath_property (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*, uchr_t * *name*, uchr_t * *val*)

Set the value of a specified property of a node element selected by an xpath expression.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

name a null-terminated string containing the name of the node's property to be set.

val the new value of the selected node's property, as a null-terminated string.

Returns:

true if the selected node's value was set successfully, or false on failure.

Definition at line 735 of file xml.c.

References log_pedantic, xml_free_xpath_obj(), xml_node_set_property(), and xml_xpath_eval().

Referenced by contact_print_form(), contact_print_message(), and teacher_print_form().

5.60.1.39 bool_t xml_set_xpath_uint64 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*, uint64_t *val*)

Set the value of a node element selected by an xpath expression, as a 64-bit unsigned integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

val the new value of the selected node, as a 64-bit unsigned integer.

Returns:

true if the value was set successfully, or false on failure.

Definition at line 668 of file xml.c.

References log_pedantic, xml_free_xpath_obj(), xml_node_set_content(), and xml_xpath_eval().

Referenced by statistics_process().

5.60.1.40 bool_t xml_start (void)

Initialize the xml parser library.

Returns:

This function returns no value.

Definition at line 1079 of file xml.c.

References xmlInitParser_d.

Referenced by process_start().

5.60.1.41 void xml_stop (void)

Cleanup the state and allocated memory of the xml parser in preparation to be shutdown.

Returns:

This function returns no value.

Definition at line 1066 of file xml.c.

References xmlCleanupGlobals_d, xmlCleanupParser_d, and xmlMemoryDump_d.

Referenced by process_stop().

5.60.1.42 xmlXPathObjectPtr xml_xpath_eval (const uchr_t * *xpath*, xmlXPathContextPtr *ctx*)

Evaluate an xml xpath expression.

Parameters:

a null-terminated string containing the xpath expression to be evaluated.

a pointer to the xpath context in which to perform the evaluation. NULL on failure, or a pointer to the xml path object of the evaluation on success.

Definition at line 941 of file xml.c.

References log_pedantic, and xmlXPathEvalExpression_d.

Referenced by contact_business_add_error(), portal_print_login(), teacher_add_error(), xml_set_xpath_ns(), xml_set_xpath_property(), and xml_set_xpath_uint64().

5.60.1.43 int_t xml_xpath_set_namespace (xmlXPathContextPtr *ctxt*, xmlChar * *prefix*, xmlChar * *ns_uri*)

Set the namespace for an xpath context.

See also:

xmlXPathRegisterNs()

Parameters:

ctxt a pointer to the specified xpath context.

prefix a pointer to the prefix of the namespace as a string.

ns_uri a pointer to the uri of the namespace as a string.

Returns:

-1 on failure or 0 on success.

Definition at line 82 of file xml.c.

References xmlXPathRegisterNs_d.

Referenced by http_page_get().

5.61 src/core/parsers/special/bracket.c File Reference

```
#include "magma.h"
```

Functions

- [placer_t bracket_extract_pl](#) (void **block*, size_t [length](#))
Get a pointer to a block of data between a pair of brackets.

5.61.1 Function Documentation

5.61.1.1 [placer_t bracket_extract_pl](#) (void * *block*, size_t *length*)

Get a pointer to a block of data between a pair of brackets.

Note:

The data is of format "[_data_]"

Parameters:

block a buffer containing the bracketed data.

length the length, in bytes, of the data buffer to be parsed.

Returns:

a null placer on failure, or a placer pointing to the data between the brackets on success.

Definition at line 17 of file bracket.c.

References [data](#), [log_pedantic](#), [pl_init\(\)](#), and [pl_null\(\)](#).

Referenced by [cache_config\(\)](#), [relay_config\(\)](#), and [servers_config\(\)](#).

5.62 src/core/parsers/special/special.h File Reference

Functions

- `placer_t bracket_extract_pl` (void **block*, size_t *length*)
Get a pointer to a block of data between a pair of brackets.

5.62.1 Function Documentation

5.62.1.1 `placer_t bracket_extract_pl` (void * *block*, size_t *length*)

Get a pointer to a block of data between a pair of brackets.

Note:

The data is of format "[_data_]"

Parameters:

block a buffer containing the bracketed data.

length the length, in bytes, of the data buffer to be parsed.

Returns:

a null placer on failure, or a placer pointing to the data between the brackets on success.

Definition at line 17 of file bracket.c.

References `data`, `log_pedantic`, `pl_init()`, and `pl_null()`.

Referenced by `cache_config()`, `relay_config()`, and `servers_config()`.

5.63 src/core/parsers/time.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t time_till_midnight (void)`
Get the number of seconds until midnight.
- `uint64_t time_datestamp (void)`
Get the current date as an integer of the form YYYYMMDD.
- `stringer_t * time_print_local (stringer_t *s, chr_t *format, time_t moment)`
Get a specified time as a formatted string.
- `stringer_t * time_print_gmt (stringer_t *s, chr_t *format, time_t moment)`
Get a specified time as a formatted string converted to GMT.

5.63.1 Function Documentation

5.63.1.1 uint64_t time_datestamp (void)

Get the current date as an integer of the form YYYYMMDD. [time.c](#)

Returns:

0 on failure or a 64-bit unsigned integer containing the formatted current date on success.

Definition at line 36 of file `time.c`.

Referenced by `api_endpoint_auth()`, `imap_login()`, `log_rotate()`, `log_start()`, `pattern_update()`, `pop_pass()`, `portal_endpoint_auth()`, `process_maint()`, `smtp_auth_login()`, and `smtp_auth_plain()`.

5.63.1.2 stringer_t* time_print_gmt (stringer_t *s, chr_t *format, time_t moment)

Get a specified time as a formatted string converted to GMT.

Parameters:

- s* a managed string where the formatted time is to be stored.
- format* a null-terminated string containing the format of the time string.
- moment* the specified time to be formatted, as a UNIX time value.

Returns:

NULL on failure, or a pointer to the managed string containing the result on success.

Definition at line 92 of file `time.c`.

References `log_pedantic`, `mm_wipe()`, `st_avail_get()`, `st_char_get()`, and `st_length_set()`.

Referenced by `http_response_cookie()`, `http_response_header()`, and `http_response_options()`.

5.63.1.3 `stringer_t* time_print_local (stringer_t * s, chr_t * format, time_t moment)`

Get a specified time as a formatted string.

Parameters:

s a managed string where the formatted time is to be stored.

format a null-terminated string containing the format of the time string.

moment the specified time to be formatted, as a UNIX time value.

Returns:

NULL on failure, or a pointer to the managed string containing the result on success.

Definition at line 61 of file time.c.

References `log_pedantic`, `mm_wipe()`, `st_avail_get()`, `st_char_get()`, and `st_length_set()`.

Referenced by `imap_id()`, and `register_data_insert_user()`.

5.63.1.4 `uint64_t time_till_midnight (void)`

Get the number of seconds until midnight.

Returns:

an unsigned 64-bit integer containing the number of seconds until midnight, or 86400 on failure.

Definition at line 14 of file time.c.

Referenced by `process_maint()`.

5.64 src/core/parsers/token.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t tok_get_count_bl` (void *block, size_t length, char token)
Count the number of times a token is found in a specified block of memory.
- `uint64_t tok_get_count_st` (stringer_t *string, char token)
Count the number of times a token is found in a managed string.
- `int tok_get_ns` (char *string, size_t length, char token, uint64_t fragment, placer_t *value)
Retrieve a specified token from a null-terminated string.
- `int tok_get_bl` (void *block, size_t length, char token, uint64_t fragment, placer_t *value)
Retrieve a specified token from a block of data.
- `int tok_get_st` (stringer_t *string, char token, uint64_t fragment, placer_t *value)
Retrieve a specified token from a managed string.
- `int tok_get_pl` (placer_t string, char token, uint64_t fragment, placer_t *value)
Retrieve a specified token from a placer.
- `void tok_pop_init_bl` (tok_state_t *state, void *block, size_t length, char token)
- `void tok_pop_init_st` (tok_state_t *state, stringer_t *string, char token)
- `int tok_pop` (tok_state_t *state, placer_t *value)
- `uint64_t str_tok_get_count_bl` (void *block, size_t length, chr_t *token, size_t tokenlen)
Count the number of times a string token is found in a specified block of memory.
- `int str_tok_get_bl` (char *block, size_t length, chr_t *token, size_t tokenlen, uint64_t fragment, placer_t *value)
Retrieve a specified string-split token from a null-terminated string.
- `bool_t pl_skip_characters` (placer_t *place, char *skipchars, size_t nchars)
Skip past any of the specified characters found at the beginning of the placer, and update the placer accordingly.
- `bool_t pl_skip_to_characters` (placer_t *place, char *skiptochars, size_t nchars)
Skip to the first instance of any of the specified characters in the placer, and update the placer accordingly.
- `bool_t pl_shrink_before_characters` (placer_t *place, char *shrinkchars, size_t nchars)
Truncate a placer to start before any of the specified characters, and update the placer accordingly.
- `bool_t pl_get_embraced` (placer_t str, placer_t *out, unsigned char opening, unsigned char closing, bool_t required)
Get a placer pointing to the text contained within a specified pair of opening and closing braces.
- `bool_t pl_update_start` (placer_t *place, size_t nchars, bool_t more)
Ensure that a placer contains at least a certain amount of data, and update it accordingly.

5.64.1 Function Documentation

5.64.1.1 `bool_t pl_get_embraced (placer_t str, placer_t * out, unsigned char opening, unsigned char closing, bool_t required)`

Get a placer pointing to the text contained within a specified pair of opening and closing braces.

Parameters:

- str* a placer pointing to the string to be parsed.
- out* a pointer to a placer that will store the string between the braces on success.
- opening* the character to be the opening brace of the sequence.
- closing* the character to be the closing brace of the sequence.
- required* if true, fail if no data was found between the pair of braces.

Returns:

true if the brace sequence was complete, or false if either component could not be located.

Definition at line 442 of file token.c.

References `pl_char_get()`, and `pl_empty()`.

5.64.1.2 `bool_t pl_shrink_before_characters (placer_t * place, char * shrinkchars, size_t nchars)`

Truncate a placer to start before any of the specified characters, and update the placer accordingly.

Parameters:

- place* a pointer to a placer that will be updated to be truncated before any of the specified characters.
- shrinkchars* a pointer to a buffer containing bytes that will be skipped when they are found at the end of the placer.
- nchars* the number of characters to be tested in the collection in *shrinkchars*.

Returns:

true if the shrink operation completed before the end of the placer was reached, or false otherwise.

Definition at line 405 of file token.c.

References `pl_char_get()`, `pl_empty()`, and `pl_length_get()`.

Referenced by `portal_upload()`.

5.64.1.3 `bool_t pl_skip_characters (placer_t * place, char * skipchars, size_t nchars)`

Skip past any of the specified characters found at the beginning of the placer, and update the placer accordingly.

Parameters:

- place* a pointer to a placer that will be updated to skip past any of the specified characters.
- skipchars* a pointer to a buffer containing bytes that will be skipped at the beginning of the placer.
- nchars* the number of characters to be tested in the collection in *skipchars*.

Returns:

true if the skip operation completed before the end of the placer was reached, or false otherwise.

Definition at line 336 of file token.c.

References `pl_char_get()`, and `pl_empty()`.

Referenced by `portal_upload()`.

5.64.1.4 bool_t pl_skip_to_characters (placer_t * *place*, char * *skiptochars*, size_t *nchars*)

Skip to the first instance of any of the specified characters in the placer, and update the placer accordingly.

Parameters:

place a pointer to a placer that will be updated to skip to any of the specified characters.

skiptochars a pointer to a buffer containing bytes that will be skipped to when they are first found in the placer.

nchars the number of characters to be tested in the collection in *skiptochars*.

Returns:

true if the skip operation completed before the end of the placer was reached, or false otherwise.

Definition at line 372 of file token.c.

References `pl_char_get()`, and `pl_empty()`.

Referenced by `portal_upload()`.

5.64.1.5 bool_t pl_update_start (placer_t * *place*, size_t *nchars*, bool_t *more*)

Ensure that a placer contains at least a certain amount of data, and update it accordingly.

Parameters:

place a pointer to the placer containing the original data, which will be updated on success.

nchars the minimum number of characters that the placer needs to contain to pass the test.

more if true, more characters following the minimum buffer size are necessary.

Returns:

true if the placer meets the minimum size check, or false if it does not.

Definition at line 485 of file token.c.

5.64.1.6 int str_tok_get_bl (char * *block*, size_t *length*, chr_t * *token*, size_t *toklen*, uint64_t *fragment*, placer_t * *value*)

Retrieve a specified string-split token from a null-terminated string.

Parameters:

block a pointer to the block of memory to be tokenized.

length the maximum number of characters to be scanned from the input data.

token the token string that will be used to split the data.

toklen the length, in bytes, of the token string.

fragment the zero-indexed token number to be extracted from the data.

value a pointer to a placer that will receive the value of the extracted token on success, or `pl_null()` on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one in the string.

Definition at line 290 of file token.c.

References `mm_empty()`, `pl_char_get()`, `pl_init()`, `pl_length_get()`, `pl_null()`, `placer_t`, and `st_search_cs()`.

Referenced by `portal_upload()`.

5.64.1.7 `uint64_t str_tok_get_count_bl (void * block, size_t length, chr_t * token, size_t toklen)`

Count the number of times a string token is found in a specified block of memory.

Parameters:

- block*** a pointer to a block of memory to be scanned.
- length*** the length, in bytes, of the block of memory to be scanned.
- token*** a pointer to the string token being used to split the input data.
- toklen*** the length, in bytes, of the specified token.

Returns:

the number of times the string token was found in the block of memory, or 0 on failure.

Definition at line 257 of file token.c.

References `count`, `mm_empty()`, `pl_init()`, `pl_length_get()`, `placer_t`, and `st_search_cs()`.

Referenced by `portal_upload()`.

5.64.1.8 `int tok_get_bl (void * block, size_t length, char token, uint64_t fragment, placer_t * value)`

Retrieve a specified token from a block of data.

See also:

[tok_get_ns\(\)](#)

Parameters:

- block*** a pointer to the memory block to be tokenized.
- length*** the maximum number of characters to be scanned at the input data block.
- token*** the token character that will be used to split the data block.
- fragment*** the zero-indexed token number to be extracted from the data block.
- value*** a pointer to a placer that will receive the value of the extracted token on success, or [pl_null\(\)](#) on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one found in the data block.

Definition at line 128 of file token.c.

References `tok_get_ns()`.

Referenced by `nvp_parse()`, `smtp_parse_mail_from_path()`, `tok_get_pl()`, and `tok_get_st()`.

5.64.1.9 `uint64_t tok_get_count_bl (void * block, size_t length, char token)`

Count the number of times a token is found in a specified block of memory.

Parameters:

- block*** a pointer to a block of memory to be scanned.
- length*** the length, in bytes, of the block of memory to be scanned.
- token*** a character specifying the token to be searched.

Returns:

the number of times the token was found in the block of memory, or 0 on failure.

Definition at line 17 of file token.c.

References `count`, `log_pedantic`, and `mm_empty()`.

Referenced by `smtp_parse_mail_from_path()`, and `tok_get_count_st()`.

5.64.1.10 uint64_t tok_get_count_st (stringer_t * string, char token)

Count the number of times a token is found in a managed string.

See also:

[tok_get_count_bl\(\)](#)

Parameters:

string a pointer to the managed string to be scanned.

token a character specifying the token to be searched.

Returns:

the number of times the token was found in the managed string, or 0 on failure.

Definition at line 49 of file token.c.

References `st_data_get()`, `st_length_get()`, and `tok_get_count_bl()`.

Referenced by `get_header_opt()`, `get_header_value_noopt()`, `http_parse_method()`, `http_parse_pairs()`, `imap_fetch_body_part()`, `imap_narrow_messages()`, `imap_search_messages_date()`, `imap_search_messages_date_compare()`, `imap_search_messages_range()`, `ip_subnet_st()`, `mail_domain_get()`, `mail_mime_type_parameters()`, `portal_get_upload_attachment()`, and `stamp_counter_check()`.

5.64.1.11 int tok_get_ns (char * string, size_t length, char token, uint64_t fragment, placer_t * value)

Retrieve a specified token from a null-terminated string.

Parameters:

string a pointer to the null-terminated string to be tokenized.

length the maximum number of characters to be scanned from the input string.

token the token character that will be used to split the string.

fragment the zero-indexed token number to be extracted from the string.

value a pointer to a placer that will receive the value of the extracted token on success, or [pl_null\(\)](#) on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one in the string.

Definition at line 63 of file token.c.

References `log_pedantic`, `mm_empty()`, `pl_init()`, and `pl_null()`.

Referenced by `ip_subnet_st()`, `lib_load_openssl()`, and `tok_get_bl()`.

5.64.1.12 `int tok_get_pl (placer_t string, char token, uint64_t fragment, placer_t * value)`

Retrieve a specified token from a placer.

See also:

`token_get_ns()`

Parameters:

string a pointer to the placer to be tokenized.

token the token character that will be used to split the placer.

fragment the zero-indexed token number to be extracted from the placer.

value a pointer to a placer that will receive the value of the extracted token on success, or `pl_null()` on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one found in the placer.

Definition at line 156 of file token.c.

References `pl_data_get()`, `pl_length_get()`, and `tok_get_bl()`.

Referenced by `http_data_value_parse()`, `http_parse_context()`, `http_parse_method()`, and `smtp_parse_mail_from_path()`.

5.64.1.13 `int tok_get_st (stringer_t * string, char token, uint64_t fragment, placer_t * value)`

Retrieve a specified token from a managed string.

See also:

`tok_get_ns()`

Parameters:

string a pointer to the managed string to be tokenized.

token the token character that will be used to split the managed string.

fragment the zero-indexed token number to be extracted from the managed string.

value a pointer to a placer that will receive the value of the extracted token on success, or `pl_null()` on failure.

Returns:

-1 on failure, 0 on success, or 1 if the token was extracted successfully, but was the last one found in the managed string.

Definition at line 142 of file token.c.

References `st_data_get()`, `st_length_get()`, and `tok_get_bl()`.

Referenced by `build_version_major()`, `build_version_minor()`, `build_version_patch()`, `get_header_opt()`, `get_header_value_noopt()`, `http_parse_context()`, `http_parse_pairs()`, `imap_fetch_body_part()`, `imap_folder_create()`, `imap_folder_remove()`, `imap_folder_rename()`, `imap_narrow_messages()`, `imap_parse_address_part()`, `imap_search_messages_date()`, `imap_search_messages_date_compare()`, `imap_search_messages_range()`, `mail_domain_get()`, `mail_mime_type_parameters()`, `portal_get_upload_attachment()`, `smtp_auth_plain()`, and `stamp_counter_check()`.

5.64.1.14 `int tok_pop (tok_state_t * state, placer_t * value)`

on error the function returns -1 and sets value to `EMPTY_PLACER` on success the function returns 0 and stores the token in the placer pointed at by value if the end is hit, then the function returns 1, and places any remaining data in value

Definition at line 188 of file token.c.

References `pl_init()`, `pl_null()`, `tok_state_t::position`, `tok_state_t::remaining`, and `tok_state_t::token`.

Referenced by `nvp_parse()`, and `prime_pem_unwrap()`.

5.64.1.15 `void tok_pop_init_bl(tok_state_t * state, void * block, size_t length, char token)`

Definition at line 161 of file token.c.

References `tok_state_t::block`, `tok_state_t::length`, `tok_state_t::position`, `tok_state_t::remaining`, and `tok_state_t::token`.

5.64.1.16 `void tok_pop_init_st(tok_state_t * state, stringer_t * string, char token)`

Definition at line 172 of file token.c.

References `tok_state_t::block`, `tok_state_t::length`, `tok_state_t::position`, `tok_state_t::remaining`, `st_char_get()`, `st_length_get()`, and `tok_state_t::token`.

Referenced by `nvp_parse()`, and `prime_pem_unwrap()`.

5.65 src/core/parsers/trim.c File Reference

```
#include "magma.h"
```

Functions

- void [st_trim](#) ([stringer_t](#) *string)
- [placer_t pl_trim](#) ([placer_t](#) place)
trim.c
- [placer_t pl_trim_start](#) ([placer_t](#) place)
Trim the leading whitespace from a placer.
- [placer_t pl_trim_end](#) ([placer_t](#) place)

5.65.1 Function Documentation

5.65.1.1 [placer_t pl_trim](#) ([placer_t](#) *place*)

[trim.c](#)

Definition at line 42 of file [trim.c](#).

References [pl_char_get\(\)](#), [pl_init\(\)](#), [pl_length_get\(\)](#), and [pl_null\(\)](#).

Referenced by [nvp_parse\(\)](#), [prime_pem_unwrap\(\)](#), and [smtp_parse_mail_from_path\(\)](#).

5.65.1.2 [placer_t pl_trim_end](#) ([placer_t](#) *place*)

Definition at line 84 of file [trim.c](#).

References [pl_char_get\(\)](#), [pl_init\(\)](#), [pl_length_get\(\)](#), and [pl_null\(\)](#).

Referenced by [smtp_parse_helo_domain\(\)](#), [smtp_parse_mail_from_path\(\)](#), and [smtp_parse_rcpt_to\(\)](#).

5.65.1.3 [placer_t pl_trim_start](#) ([placer_t](#) *place*)

Trim the leading whitespace from a placer.

Parameters:

place a placer containing the string to have its leading whitespace trimmed.

Returns:

a placer pointing to the trimmed value inside the originally specified input string.

Definition at line 67 of file [trim.c](#).

References [pl_char_get\(\)](#), [pl_init\(\)](#), [pl_length_get\(\)](#), and [pl_null\(\)](#).

Referenced by [get_header_opt\(\)](#).

5.65.1.4 void [st_trim](#) ([stringer_t](#) * *string*)

Definition at line 15 of file [trim.c](#).

References `mm_move()`, `mm_wipe()`, `st_avail_get()`, `st_char_get()`, `st_length_get()`, and `st_length_set()`.

Referenced by `mail_header_fetch_cleaned()`, and `process_find_pid()`.

5.66 src/core/strings/allocation.c File Reference

```
#include "magma.h"
```

Functions

- `void st_free (stringer_t *s)`
Free a managed string.
- `void st_cleanup_variadic (ssize_t len,...)`
A checked cleanup function which can be used free a variable number managed strings.
- `stringer_t * st_merge_opts (uint32_t opts, chr_t *format,...)`
Concatenate a variable number of user-supplied multi-type strings.
- `stringer_t * st_import_opts (uint32_t opts, const void *s, size_t len)`
Create a new (contiguous managed) managed string on the heap to hold a copy of the specified data buffer.
- `stringer_t * st_import (const void *s, size_t len)`
Create a new (contiguous managed) managed string on the heap to hold a copy of the specified data buffer.
- `stringer_t * st_copy_in (stringer_t *s, void *buf, size_t len)`
Copy data into a managed string.
- `stringer_t * st_dupe_opts (uint32_t opts, stringer_t *s)`
Create a duplicate copy of a managed string with a specified set of allocation options.
- `stringer_t * st_dupe (stringer_t *s)`
Duplicate a managed string.
- `stringer_t * st_append_opts (size_t align, stringer_t *s, stringer_t *append)`
Append one managed string to another, with aligned memory boundaries.
- `int_t st_append_out (size_t align, stringer_t **s, stringer_t *append)`
- `stringer_t * st_alloc_opts (uint32_t opts, size_t len)`
Allocate a managed string with a specified options mask.
- `stringer_t * st_alloc (size_t len)`
Allocate a managed string with a specified options mask.
- `stringer_t * st_realloc (stringer_t *s, size_t len)`
Reallocate a managed string to a specified size.
- `stringer_t * st_output (stringer_t *output, size_t len)`
Return a suitable managed string to store data of a specified length.
- `stringer_t * st_nullify (chr_t *input, size_t len)`
Create a null-terminated string out of a block of memory and return it as a newly allocated nuller.

5.66.1 Function Documentation

5.66.1.1 `stringer_t* st_alloc (size_t len)`

Allocate a managed string with a specified options mask.

See also:

[st_alloc_opts\(\)](#)

Parameters:

len the length, in bytes, of the managed string to be allocated.

Returns:

NULL on failure or a pointer to the newly allocated managed string on success.

Definition at line 631 of file allocation.c.

References CONTIGUOUS, HEAP, MANAGED_T, and st_alloc_opts().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), auth_legacy(), auth_sanitizer_address(), base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), client_connect(), con_init_network_buffer(), decompress_block_lzo(), decompress_bzip(), decompress_lzo(), decompress_zlib(), deprecated_hmac_digest(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), deserialize_st(), ed25519_private_get(), ed25519_public_get(), ed25519_sign(), encrypted_chunk_set(), file_load(), hash_digest(), hex_decode_st(), hex_encode_st(), hex_encode_st_debug(), hmac_digest(), http_load_file(), imap_build_array(), imap_parse_address_part(), imap_parse_literal(), imap_parse_qstring(), ip_presentation(), ip_reversed(), ip_standard(), ip_subnet(), mail_add_required_headers(), mail_build_signature(), mail_extract_address(), mail_extract_tag(), mail_header_fetch_cleaned(), mail_load_message(), mail_mime_generate_boundary(), org_key_get(), org_signet_get(), portal_smtp_merge_headers(), qp_decode(), rand_choices(), sanity_check(), secp256k1_compute_kek(), secp256k1_private_get(), secp256k1_public_get(), serialize_int16(), serialize_int32(), serialize_int64(), serialize_uint16(), serialize_uint32(), serialize_uint64(), signature_full_get(), signature_tree_get(), st_bitwise(), st_not(), st_output(), st_replace(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_decrypt(), stacie_encrypt(), symmetric_decrypt(), symmetric_encrypt(), uint16_put_no(), uint24_put_no(), uint32_put_no(), url_decode(), user_key_get(), user_request_get(), user_signet_get(), zbase32_decode(), and zbase32_encode().

5.66.1.2 `stringer_t* st_alloc_opts (uint32_t opts, size_t len)`

Allocate a managed string with a specified options mask.

See also:

[st_valid_options\(\)](#)

Note:

All requested allocation masks should conform to the validation imposed by [st_valid_opts\(\)](#). The supported types are: placer, nuller, block, managed, and mapped. The following logic is applied to requested string allocation options: 1. Any allocation options specified for strings to be allocated on the stack are IGNORED. 2. All jointed strings allocate memory for the header and data separately and then link them EXCEPT: Jointed placers only allocate space for a header. Jointed mapped strings allocate the data with an aligned mmap() operation. 3. Contiguous strings allocate space for the header and data together in a single contiguous block. 4. After allocation, the length field is set for block strings. 5. After allocation, the available field is set for managed strings. 6. Placers can only be jointed; mapped strings can only be contiguous.

Parameters:

opts the allocation options mask to be used by the newly allocated string.

len the length, in bytes, of the managed string to be allocated.

Returns:

NULL on failure or a pointer to the newly allocated managed string on success.

Definition at line 498 of file allocation.c.

References align(), block_t, BLOCK_T, CONTIGUOUS, HEAP, JOINTED, log_pedantic, magma, MAGMA_SPOOL_DATA, managed_t, MANAGED_T, mapped_t, MAPPED_T, MEMORYBUF, mm_alloc(), mm_free(), mm_sec_alloc(), mm_sec_free(), mm_set(), nuller_t, NULLER_T, magma_t::page_length, placer_t, PLACER_T, SECURE, spool_mktemp(), st_info_opts(), st_valid_opts(), and STACK.

Referenced by auth_legacy(), base64_decode_opts(), base64_encode_opts(), dkim_signature_create(), ephemeral_chunk_set(), hex_decode_opts(), hex_encode_opts(), http_body(), mail_add_forward_headers(), mail_message_cleanup(), mail_mime_split(), naked_message_set(), part_decrypt(), pop_pass_parse(), qp_encode(), signature_tree_get(), signature_tree_verify(), smtp_check_greylist(), smtp_data_read(), st_alloc(), st_append_opts(), st_append_out(), st_dupe_opts(), st_import_opts(), st_merge_opts(), st_nullify(), st_vaprint_opts(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_realm_key(), and url_encode().

5.66.1.3 stringer_t* st_append_opts (size_t align, stringer_t * s, stringer_t * append)

Append one managed string to another, with aligned memory boundaries.

Parameters:

align an alignment value to be used for managed string (re)allocation. This only works for powers of 2.

s the managed string to be extended, which will be allocated for the caller if passed as NULL.

append the managed string to be appended to *s*.

Returns:

NULL on failure, or a pointer to the appended result on success.

Definition at line 407 of file allocation.c.

References HEAP, JOINTED, log_pedantic, MANAGED_T, mm_copy(), st_alloc_opts(), st_avail_get(), st_data_get(), st_length_get(), st_length_set(), st_realloc(), and st_valid_append().

Referenced by http_body(), imap_id(), imap_search(), imap_status(), and mail_add_forward_headers().

5.66.1.4 int_t st_append_out (size_t align, stringer_t ** s, stringer_t * append)

Definition at line 439 of file allocation.c.

References HEAP, JOINTED, log_pedantic, MANAGED_T, mm_copy(), st_alloc_opts(), st_avail_get(), st_data_get(), st_free(), st_length_get(), st_length_set(), st_realloc(), and st_valid_destination().

Referenced by encrypted_chunk_set().

5.66.1.5 void st_cleanup_variadic (ssize_t len, ...)

A checked cleanup function which can be used free a variable number managed strings.

See also:

[st_free\(\)](#)

Parameters:

va_list a list of managed strings to be freed.

Returns:

This function returns no value.

Definition at line 96 of file allocation.c.

References `st_free()`.

5.66.1.6 `stringer_t* st_copy_in (stringer_t * s, void * buf, size_t len)`

Copy data into a managed string.

Parameters:

- s* the managed string to store the copied contents of the data.
- buf* a pointer to the buffer containing the data to be copied.
- len* the length of the data to be copied.

Returns:

NULL on failure, or a pointer to the managed string on success.

Definition at line 279 of file allocation.c.

References `log_pedantic`, `mm_copy()`, `st_avail_get()`, `st_data_get()`, and `st_length_set()`.

Referenced by `pop_pass_parse()`.

5.66.1.7 `stringer_t* st_dupe (stringer_t * s)`

Duplicate a managed string.

See also:

[st_dupe_opts\(\)](#)

Note:

The allocation options of the duplicated string will be the same as that of the source string.

Parameters:

- s* the managed string to be duplicated.

Returns:

NULL on failure, or a copy of the input managed string on success.

Definition at line 393 of file allocation.c.

References `st_dupe_opts()`.

Referenced by `ar_dupe()`, `auth_login()`, `auth_response()`, `contact_folder_rename()`, `file_temp_handle()`, `http_load_file()`, `http_print_301()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_login()`, `mail_add_forward_headers()`, `mail_add_outbound_headers()`, `mail_cache_get()`, `pop_pass()`, `prime_get()`, `prime_key_encrypt()`, `prime_pem_wrap()`, `register_business_step1()`, `register_business_step2()`, `register_print_message()`, `register_print_step1()`, `register_print_step2()`, `register_print_step3()`, `register_session_get()`, `smtp_add_recipient()`, `smtp_data_outbound()`, `smtp_fetch_inbound()`, `teacher_process()`, and `user_request_sign()`.

5.66.1.8 `stringer_t* st_dupe_opts (uint32_t opts, stringer_t * s)`

Create a duplicate copy of a managed string with a specified set of allocation options.

See also:

[st_valid_opts\(\)](#)

Note:

Both the source and destination managed strings must have option values that pass [st_valid_opts\(\)](#). The following procedure occurs when creating the duplicate managed string: If the destination is a placer, 0 bytes are allocated for the duplicate's underlying data. If the destination is a constant, nuller, or block, its underlying data buffer will be of equal size to the source's data length. If the destination is managed or mapped, and so is the source, its underlying data buffer will be of equal size to the source's available size; but if the source is neither, the underlying data buffer of the destination managed or mapped string will equal the size of the source's data length. A deep copy will be made of the source data to the destination if the destination is NOT a placer.

Parameters:

opts the allocation options mask to be used for the newly created managed string.
s the target managed string to be duplicated.

Returns:

NULL on failure or a pointer to a copy of the duplicated managed string on success.

Definition at line 319 of file allocation.c.

References [BLOCK_T](#), [CONSTANT_T](#), [log_pedantic](#), [MANAGED_T](#), [MAPPED_T](#), [MEMORYBUF](#), [mm_copy\(\)](#), [NULLER_T](#), [PLACER_T](#), [st_alloc_opts\(\)](#), [st_avail_get\(\)](#), [st_data_get\(\)](#), [st_data_set\(\)](#), [st_info_opts\(\)](#), [st_length_get\(\)](#), [st_length_set\(\)](#), [st_valid_opts\(\)](#), and [st_valid_tracked\(\)](#).

Referenced by [args_parse\(\)](#), [cache_set_value\(\)](#), [config_value_set\(\)](#), [file_temp_handle\(\)](#), [http_body\(\)](#), [http_data_value_parse\(\)](#), [http_load_file\(\)](#), [http_parse_header\(\)](#), [http_parse_method\(\)](#), [http_response_connection\(\)](#), [imap_fetch_response_add\(\)](#), [imap_folder_create\(\)](#), [imap_folder_rename\(\)](#), [magma_folder_name\(\)](#), [mail_add_forward_headers\(\)](#), [mail_cache_set\(\)](#), [mt_dupe\(\)](#), [portal_message_body\(\)](#), [relay_set_value\(\)](#), [servers_set_value\(\)](#), [sess_create\(\)](#), [smtp_client_send_data\(\)](#), [smtp_fetch_inbound\(\)](#), [smtp_forward_message\(\)](#), [smtp_relay_message\(\)](#), [st_dupe\(\)](#), [stacie_realm_cipher\(\)](#), [stacie_realm_tag\(\)](#), and [stacie_realm_vector\(\)](#).

5.66.1.9 void st_free (stringer_t * s)

Free a managed string.

See also:

[st_valid_free\(\)](#), [st_valid_opts\(\)](#)

Note:

Any managed string to be freed should conform with the validity checks enforced by [st_valid_free\(\)](#)/[st_valid_opts\(\)](#) *NO* managed string should be passed to [st_free\(\)](#) if it was allocated on the stack, or contains foreign data. The following logic is carried out depending on the allocation options of the string to be freed: Jointed placer: Free header, if secure or on heap Free underlying data if it is not foreign Jointed nuller: Free header and underlying data Jointed block: Free header and underlying data Jointed managed: Free header and underlying data Jointed mapped: Free the header and (munmap) underlying data Contiguous nuller: Free header (this includes the underlying data because they are already merged) Contiguous block: Free header (this includes the underlying data because they are already merged) Contiguous managed: Free header (this includes the underlying data because they are already merged)

Parameters:

s the managed string to be freed.

Returns:

This function returns no value.

HIGH: Finish this logic out. Stack allocations are skipped, unless they are jointed. In that case the data is freed unless it carries a foreigner flag. Note its possible for a jointed string to have a NULL data reference. We should be picking up on that too. Contiguous heap buffers are just destroyed.

Do we need to differentiate between stack structures, and heap data blocks needing to be freed, or not?

Definition at line 29 of file allocation.c.

References `block_t`, `BLOCK_T`, `CONTIGUOUS`, `data`, `FOREIGNDATA`, `HEAP`, `JOINTED`, `log_pedantic`, `managed_t`, `MANAGED_T`, `mapped_t`, `MAPPED_T`, `MEMORYBUF`, `mm_free()`, `mm_sec_free()`, `nuller_t`, `NULLER_T`, `placer_t`, `PLACER_T`, `SECURE`, `st_info_opts()`, and `st_valid_free()`.

Referenced by `ar_free()`, `auth_data_fetch()`, `auth_legacy()`, `auth_response()`, `base64_decode_opts()`, `base64_encode_opts()`, `cache_free()`, `cache_set_value()`, `client_print()`, `compress_bzip()`, `compress_lzo()`, `compress_zlib()`, `config_free()`, `config_load_cmdline_settings()`, `config_load_file_settings()`, `config_value_set()`, `contact_business()`, `contact_folder_rename()`, `contact_free()`, `contact_print_form()`, `contact_print_message()`, `decompress_block_lzo()`, `decompress_bzip()`, `decompress_lzo()`, `decompress_zlib()`, `deprecated_hmac_digest()`, `deprecated_scramble_decrypt()`, `deprecated_scramble_encrypt()`, `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `deprecated_symmetric_vector()`, `dkim_start()`, `dspam_check()`, `dspam_train()`, `encrypted_chunk_free()`, `encrypted_chunk_get()`, `encrypted_message_free()`, `ephemeral_chunk_free()`, `file_temp_handle()`, `folder_count()`, `hash_digest()`, `hex_decode_opts()`, `hex_encode_opts()`, `hmac_digest()`, `http_content_load_directory()`, `http_content_load_fonts()`, `http_content_refresh()`, `http_content_start()`, `http_data_header_parse_line()`, `http_data_value_parse()`, `http_load_file()`, `http_print_301()`, `imap_build_array()`, `imap_command_parser()`, `imap_fetch_body()`, `imap_fetch_body_tag()`, `imap_fetch_bodystructure()`, `imap_fetch_response_add()`, `imap_folder_create()`, `imap_folder_name_escaped()`, `imap_folder_rename()`, `imap_id()`, `imap_list()`, `imap_login()`, `imap_lsub()`, `imap_narrow_folders()`, `imap_parse_address()`, `imap_parse_address_part()`, `imap_parse_literal()`, `imap_range_build()`, `imap_search_messages_body()`, `imap_search_messages_date()`, `imap_search_messages_header()`, `imap_search_messages_text()`, `imap_status()`, `ip_presentation()`, `ip_standard()`, `ip_subnet()`, `mail_add_forward_headers()`, `mail_add_outbound_headers()`, `mail_add_required_headers()`, `mail_build_signature()`, `mail_create_message()`, `mail_discover_encoding()`, `mail_discover_insertion_point()`, `mail_discover_type()`, `mail_extract_tag()`, `mail_header_fetch_all()`, `mail_headers()`, `mail_insert_chunk_base64()`, `mail_insert_chunk_text()`, `mail_load_message()`, `mail_message_cleanup()`, `mail_mime_content_encoding()`, `mail_mime_content_id()`, `mail_mime_encode_part()`, `mail_mime_encoding()`, `mail_mime_generate_boundary()`, `mail_mime_get_smtp_envelope()`, `mail_mime_split()`, `mail_mime_type()`, `mail_mime_type_group()`, `mail_mime_type_parameters()`, `mail_mime_type_sub()`, `mail_mod_subject()`, `mail_modify_part()`, `meta_crypto_keys_create()`, `meta_data_fetch_all_tags()`, `meta_data_fetch_keys()`, `meta_folders_by_name()`, `meta_folders_name()`, `mt_free()`, `naked_message_get()`, `naked_message_set()`, `nvp_alloc()`, `nvp_parse()`, `org_encrypted_key_get()`, `org_encrypted_key_set()`, `org_signet_free()`, `org_signet_generate()`, `part_decrypt()`, `pop_pass()`, `portal_endpoint_attachments_progress()`, `portal_endpoint_folders_add()`, `portal_endpoint_folders_rename()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_send()`, `portal_endpoint_search()`, `portal_parse_json_str_array()`, `portal_print_login()`, `portal_smtp_create_data()`, `prime_field_write()`, `prime_get()`, `prime_header_write()`, `prime_key_encrypt()`, `prime_pem_unwrap()`, `prime_pem_wrap()`, `prime_start()`, `protocol_secure()`, `register_data_fetch_blocklist()`, `register_print_captcha()`, `register_print_message()`, `register_print_step1()`, `register_print_step2()`, `register_print_step3()`, `register_session_cache()`, `register_session_get()`, `relay_free()`, `relay_set_value()`, `sanity_check()`, `scramble_decrypt()`, `scramble_encrypt()`, `serial_get()`, `serial_increment()`, `serial_reset()`, `servers_free()`, `servers_set_value()`, `sess_get()`, `signature_tree_add()`, `signature_tree_alloc()`, `signature_tree_get()`, `signature_tree_verify()`, `slots_key()`, `smtp_accept_message()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_bounce()`, `smtp_check_receive_quota()`, `smtp_data()`, `smtp_data_outbound()`, `smtp_data_read()`, `smtp_fetch_authorization()`, `smtp_forward_message()`, `smtp_mail_from()`, `smtp_rcpt_to()`, `smtp_reply()`, `smtp_rollout()`, `smtp_update_receive_stats()`, `spool_cleanup()`, `spool_mktemp()`, `spool_start()`, `spool_stop()`, `st_append_out()`, `st_bitwise()`, `st_cleanup_variadic()`, `st_not()`, `st_replace()`, `st_vaprint_opts()`, `stacie_decrypt()`, `stacie_encrypt()`, `stamp_counter_check()`, `statistics_process()`, `symmetric_decrypt()`, `symmetric_encrypt()`, `symmetric_vector()`, `teacher_add_cookie()`, `teacher_data_get()`, `teacher_data_save()`, `teacher_print_form()`, `teacher_print_message()`, `user_encrypted_key_get()`, `user_encrypted_key_set()`, `user_signet_free()`, `virus_stop()`, `warehouse_fetch_patterns()`, and `zbase32_decode()`.

5.66.1.10 `stringer_t* st_import (const void *s, size_t len)`

Create a new (contiguous managed) managed string on the heap to hold a copy of the specified data buffer.

See also:

[st_import_opts\(\)](#)

Parameters:

- s* the address of the buffer to be duplicated.
- len* the length, in bytes, of the copied buffer.

Returns:

- NULL on failure, or a pointer to the newly allocated managed string on success.

Definition at line 267 of file allocation.c.

References CONTIGUOUS, HEAP, MANAGED_T, and st_import_opts().

Referenced by cache_get(), con_reverse_lookup(), deprecated_ecies_key_public_hex(), dspam_check(), ecies_key_public_hex(), file_temp_handle(), http_data_header_parse_line(), http_load_file(), imap_fetch_body(), imap_fetch_body_tag(), imap_fetch_bodystructure(), imap_fetch_message(), imap_fetch_return_header(), imap_parse_astring(), imap_parse_nstring(), imap_search_messages_date(), mail_add_outbound_headers(), mail_add_required_headers(), mail_header_fetch_all(), mail_load_header(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_type_group(), mail_mime_type_parameters_key(), mail_mime_type_parameters_value(), mail_mime_type_sub(), meta_folders_name(), nvp_parse(), org_signet_set(), pop_user_parse(), portal_endpoint_attachments_add(), portal_endpoint_folders_add(), portal_endpoint_folders_rename(), portal_endpoint_messages_list(), portal_parse_json_str_array(), portal_upload(), register_captcha_generate(), res_field_string(), servers_network_start(), smtp_parse_auth(), smtp_parse_helo_domain(), smtp_parse_mail_from_path(), smtp_parse_rcpt_to(), tank_load(), teacher_add_cookie(), user_request_set(), user_signet_set(), xml_dump_doc(), xml_get_xpath_st(), and xml_node_get_content_st().

5.66.1.11 stringer_t* st_import_opts (uint32_t *opts*, const void * *s*, size_t *len*)

Create a new (contiguous managed) managed string on the heap to hold a copy of the specified data buffer.

Parameters:

- opts* the options value of the resulting managed string that will be allocated for the caller.
- s* the address of the buffer to be duplicated.
- len* the length, in bytes, of the copied buffer.

Returns:

NULL on failure, or a pointer to the newly allocated managed string on success.

Definition at line 236 of file allocation.c.

References log_pedantic, MEMORYBUF, mm_copy(), st_alloc_opts(), st_data_get(), st_info_opts(), st_length_set(), and st_valid_destination().

Referenced by deprecated_ecies_key_private_hex(), ecies_key_private_hex(), and st_import().

5.66.1.12 stringer_t* st_merge_opts (uint32_t *opts*, chr_t * *format*, ...)

Concatenate a variable number of user-supplied multi-type strings.

Parameters:

- opts* the options value of the resulting managed string that will be allocated for the caller.
- format* a format string consisting of the characters 's' for specifying that the next variable argument will be a managed string or 'n' if it is a null-terminated string.
- ... a variable argument list containing the strings to be concatenated to one another.

Returns:

a newly allocated string containing the concatenations of all specified managed and null-terminated strings.

Definition at line 120 of file allocation.c.

References length, log_pedantic, MEMORYBUF, mm_copy(), ns_empty(), ns_length_get(), st_alloc_opts(), st_char_get(), st_cleanup, st_data_get(), st_empty, st_info_opts(), st_length_get(), st_length_set(), st_valid_destination(), and st_valid_tracked().

Referenced by mail_add_forward_headers(), mail_add_inbound_headers(), mail_add_outbound_headers(), mail_add_required_headers(), and spool_path().

5.66.1.13 stringer_t* st_nullify (chr_t * *input*, size_t *len*)

Create a null-terminated string out of a block of memory and return it as a newly allocated nuller.

Parameters:

input a pointer to the data block to be copied.

len the size, in bytes, of the data block to be copied before being terminated with a null byte.

Returns:

NULL on failure, or a pointer to the new null-terminated string on success.

Definition at line 841 of file allocation.c.

References CONTIGUOUS, HEAP, MANAGED_T, mm_copy(), st_alloc_opts(), st_data_get(), and st_length_set().

Referenced by portal_message_body().

5.66.1.14 stringer_t* st_output (stringer_t * *output*, size_t *len*)

Return a suitable managed string to store data of a specified length.

Note:

If a NULL output string is specified, one will be allocated for the caller.

Parameters:

output the input managed string (can be NULL).

len the size, in bytes, of the requested data buffer.

Returns:

NULL on failure or if input was not large enough or a pointer to a validated output managed string.

Definition at line 812 of file allocation.c.

References log_pedantic, st_alloc(), st_avail_get(), and st_valid_destination().

Referenced by chunk_header_write(), deprecated_symmetric_key(), deprecated_symmetric_vector(), encrypted_chunk_get(), prime_field_write(), prime_header_write(), prime_pem_wrap(), slots_key(), symmetric_key(), symmetric_vector(), tls_cipher(), and tls_error().

5.66.1.15 stringer_t* st_realloc (stringer_t * *s*, size_t *len*)

Reallocate a managed string to a specified size.

Note:

The caller can request a new size that is either smaller (truncated) or larger than the original string. Only these types of strings can be reallocated: nuller, block, managed, and mapped. The following logic is applied to string reallocation requests: For jointed strings, a new data buffer is allocated of the requested length, the original contents copied in, the data field of the new string is set, and the original data buffer freed. The stringer header returned to the caller resides at the same address as the original. For contiguous strings, a new data buffer is allocated for both the requested length and the new stringer header, and the data of both parts are copied from the original string to the resulting one. For block strings, the length field of the resulting string is set to the requested length. For managed strings, the length field of the resulting string is set to the original length, but the available field is adjusted accordingly. For mapped strings, an aligned mremap() call is made, and both the length and available fields are set to the new length.

Parameters:

s the managed string to have its size reallocated.

len the new size, in bytes, of the resulting managed string.

Returns:

NULL on failure or a pointer to the newly reallocated managed string on success.

Definition at line 656 of file allocation.c.

References `align()`, `block_t`, `BLOCK_T`, `CONTIGUOUS`, `data`, `JOINTED`, `length`, `log_pedantic`, `magma`, `managed_t`, `MANAGED_T`, `mapped_t`, `MAPPED_T`, `magma_t::memory`, `MEMORYBUF`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `mm_sec_alloc()`, `mm_sec_free()`, `nuller_t`, `NULLER_T`, `magma_t::page_length`, `magma_t::secure`, `SECURE`, `st_info_opts()`, `st_length_get()`, `st_valid_opts()`, and `system_ulimit_cur()`.

Referenced by `serialize_int16()`, `serialize_int32()`, `serialize_int64()`, `serialize_ns()`, `serialize_st()`, `serialize_uint16()`, `serialize_uint32()`, `serialize_uint64()`, `smtp_data_read()`, `st_append_opts()`, and `st_append_out()`.

5.67 src/core/strings/data.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t st_populated_variadic](#) (ssize_t len,...)
A simple method for checking multiple managed strings to ensure all of the provided strings have contain at least one character of data.
- [bool_t st_empty_variadic](#) (ssize_t len,...)
A simple method for checking multiple managed strings to see if any are empty.
- [bool_t st_empty_out](#) (stringer_t *s, uchr_t **ptr, size_t *len)
Determine whether the specified managed string is empty or not, and store its underlying data and data length.
- void [st_data_set](#) (stringer_t *s, void *data)
Set the underlying data of a jointed managed string.
- void * [st_data_get](#) (stringer_t *s)
Retrieve the data associated with a managed string.
- [chr_t * st_char_get](#) (stringer_t *s)
Retrieve a character pointer to a managed string's data buffer.
- [uchr_t * st_uchar_get](#) (stringer_t *s)
Retrieve an unsigned character pointer to a managed string's data buffer.
- void [st_wipe](#) (stringer_t *s)
Wipe all of a managed string's allocated memory and if applicable, reset its length.
- [stringer_t * st_set](#) (stringer_t *s, uint8_t set, size_t len)
Sets a block of memory to a specified value.

5.67.1 Function Documentation

5.67.1.1 chr_t* st_char_get (stringer_t * s)

Retrieve a character pointer to a managed string's data buffer. data.c

See also:

[st_data_get\(\)](#)

Parameters:

s the input managed string.

Returns:

NULL on failure or for an improperly constructed string; otherwise, a pointer to the string's data.

Definition at line 196 of file data.c.

References `st_data_get()`.

Referenced by `api_endpoint_auth()`, `api_endpoint_change_password()`, `api_endpoint_delete_user()`, `auth_data_fetch()`, `auth_data_update_legacy()`, `auth_login()`, `auth_response()`, `auth_sanitise_address()`, `auth_sanitise_username()`, `cache_add()`, `cache_append()`, `cache_config()`, `cache_decrement()`, `cache_delete()`, `cache_get()`, `cache_get_u64()`, `cache_increment()`, `cache_output_settings()`, `cache_set()`, `cache_set_u64()`, `cache_set_value()`, `cache_silent_add()`, `cipher_name()`, `client_read()`, `client_read_line()`, `client_write()`, `con_read()`, `con_read_line()`, `con_write_st()`, `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_file_settings()`, `config_output_value()`, `config_output_value_generic()`, `config_validate_settings()`, `config_value_set()`, `contact_abuse_checks()`, `contact_abuse_increment_history()`, `contact_business_valid_email()`, `contact_detail_delete()`, `contact_detail_upsert()`, `contact_insert()`, `contact_print_form()`, `contact_update()`, `deserialize_int16()`, `deserialize_int32()`, `deserialize_int64()`, `deserialize_ns()`, `deserialize_st()`, `deserialize_uint16()`, `deserialize_uint32()`, `deserialize_uint64()`, `digest_name()`, `dkim_start()`, `dsmtp_ahlo()`, `dsmtp_helo()`, `dsmtp_init()`, `double_conv()`, `dspam_check()`, `dspam_train()`, `file_temp_handle()`, `float_conv()`, `folder_count()`, `folder_exists()`, `get_header_value_noopt()`, `hex_encode_st_debug()`, `http_body()`, `http_content_load_directory()`, `http_data_header_parse()`, `http_data_value_decode()`, `http_data_value_parse()`, `http_load_file()`, `http_page_get()`, `http_parse_context()`, `http_parse_header()`, `http_parse_method()`, `http_parse_pairs()`, `http_print_301()`, `http_response_allow_cross()`, `http_response_cookie()`, `http_response_header()`, `http_response_options()`, `imap_append()`, `imap_append_message()`, `imap_build_array()`, `imap_capability()`, `imap_check()`, `imap_close()`, `imap_command_log_safe()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_fetch_body_portion()`, `imap_fetch_bodystructure()`, `imap_fetch_envelope()`, `imap_fetch_parse_partial()`, `imap_fetch_return_header()`, `imap_folder_compare()`, `imap_id()`, `imap_idle()`, `imap_init()`, `imap_invalid()`, `imap_list()`, `imap_login()`, `imap_logout()`, `imap_lsub()`, `imap_narrow_messages()`, `imap_noop()`, `imap_parse_address_breaker()`, `imap_parse_address_part()`, `imap_parse_address_put()`, `imap_parse_literal()`, `imap_parse_qstring()`, `imap_process()`, `imap_rename()`, `imap_search()`, `imap_search_messages_date()`, `imap_search_messages_header()`, `imap_select()`, `imap_starttls()`, `imap_status()`, `imap_store()`, `imap_subscribe()`, `imap_unsubscribe()`, `ip_presentation()`, `ip_reversed()`, `ip_standard()`, `json_api_dispatch()`, `lib_load()`, `line_pl_st()`, `lock_get()`, `lock_release()`, `magma_folder_insert()`, `magma_folder_rename()`, `mail_add_forward_headers()`, `mail_add_outbound_headers()`, `mail_add_required_headers()`, `mail_build_signature()`, `mail_create_directory()`, `mail_db_insert_duplicate_message()`, `mail_db_insert_message()`, `mail_discover_encoding()`, `mail_discover_insertion_point()`, `mail_discover_type()`, `mail_extract_address()`, `mail_extract_tag()`, `mail_get_boundary()`, `mail_get_chunk()`, `mail_header_end()`, `mail_header_fetch_all()`, `mail_header_fetch_cleaned()`, `mail_header_pop()`, `mail_headers()`, `mail_insert_chunk_base64()`, `mail_insert_chunk_text()`, `mail_load_header()`, `mail_load_message()`, `mail_load_message_top()`, `mail_message()`, `mail_message_cleanup()`, `mail_message_path()`, `mail_mime_boundary()`, `mail_mime_child()`, `mail_mime_content_encoding()`, `mail_mime_content_id()`, `mail_mime_count()`, `mail_mime_encoding()`, `mail_mime_generate_boundary()`, `mail_mime_header()`, `mail_mime_part()`, `mail_mime_type()`, `mail_mime_type_group()`, `mail_mime_type_parameters_key()`, `mail_mime_type_parameters_value()`, `mail_mime_type_sub()`, `mail_mime_update()`, `mail_mod_subject()`, `mail_modify_part()`, `mail_signature_add()`, `meta_crypto_keys_create()`, `meta_data_delete_tag()`, `meta_data_fetch_keys()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_shard()`, `meta_data_insert_folder()`, `meta_data_insert_keys()`, `meta_data_insert_shard()`, `meta_data_insert_tag()`, `meta_data_update_folder_name()`, `meta_update_keys()`, `meta_update_realms()`, `mt_get_char()`, `net_trigger()`, `pl_char_get()`, `pop_pass()`, `pop_retr()`, `pop_top()`, `portal_config_collection()`, `portal_config_entry()`, `portal_contact_details()`, `portal_endpoint()`, `portal_endpoint_alert_acknowledge()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_auth()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_folders_add()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_remove()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_message_attachments()`, `portal_message_body()`, `portal_message_header()`, `portal_message_tags_array()`, `portal_meta()`, `portal_settings_changepass()`, `portal_settings_identity()`, `portal_upload()`, `portal_validate_request()`, `prime_start()`, `process_kill()`, `protocol_secure()`, `register_abuse_check_blocklist()`, `register_abuse_checks()`, `register_abuse_increment_history()`, `register_business_step1()`, `register_business_validate_password()`, `register_business_validate_username()`, `register_captcha_generate()`, `register_data_check_username()`, `register_data_insert_user()`, `register_print_captcha()`, `register_session_cache()`, `register_session_get()`, `relay_config()`, `relay_output_settings()`, `relay_set_value()`, `secp256k1_public_set()`, `serial_get()`, `serial_increment()`, `serial_reset()`, `serialize_int16()`, `serialize_int32()`, `serialize_int64()`, `serialize_ns()`, `serialize_st()`, `serialize_uint16()`, `serialize_uint32()`, `serialize_uint64()`, `servers_config()`, `servers_output_settings()`, `servers_set_value()`, `sess_get()`, `smtp_add_bypass_entry()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_bounce()`, `smtp_check_authorized_from()`, `smtp_check_filters()`, `smtp_check_greylist()`, `smtp_check_rbl()`, `smtp_check_receive_quota()`, `smtp_client_connect()`, `smtp_client_send_data()`, `smtp_client_send_helo()`, `smtp_client_send_mailfrom()`, `smtp_client_send_nullfrom()`, `smtp_client_send_rcptto()`, `smtp_data_finish()`, `smtp_data_outbound()`, `smtp_data_read()`, `smtp_ahlo()`, `smtp_fetch_authorization()`, `smtp_fetch_inbound()`, `smtp_helo()`, `smtp_init()`, `smtp_insert_spamsig()`, `smtp_rcpt_to()`, `smtp_reply()`, `smtp_rollout()`, `smtp_update_receive_stats()`, `spf_check()`, `spool_check()`, `spool_cleanup()`, `spool_mktemp()`, `spool_start()`, `spool_stop()`, `sql_query_conn()`, `st_info_opts()`, `st_merge_opts()`, `st_replace()`, `st_trim()`, `stacie_derive_key()`, `stacie_derive_token()`, `teacher_add_cookie()`, `teacher_print_message()`, `teacher_process()`, `time_print_gmt()`, `time_print_local()`, `tok_pop_init_st()`, `user_config_delete()`, `user_config_upsert()`, `virus_engine_create()`, and

virus_start().

5.67.1.2 void* st_data_get (stringer_t * s)

Retrieve the data associated with a managed string.

Parameters:

s the input managed string.

Returns:

NULL on failure or for an improperly constructed string; otherwise, a pointer to the string's data.

Definition at line 149 of file data.c.

References block_t, BLOCK_T, constant_t, CONSTANT_T, log_pedantic, managed_t, MANAGED_T, mapped_t, MAPPED_T, MEMORY_BUF, nuller_t, NULLER_T, placer_t, PLACER_T, st_info_opts(), and st_valid_opts().

Referenced by _serialize_ec_pubkey(), aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), aes_cipher_key(), aes_tag_shard(), aes_vector_shard(), alert_alloc(), alias_alloc(), auth_response(), base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), chunk_header_read(), chunk_header_write(), client_read(), client_read_line(), compress_bzip(), compress_import(), compress_lzo(), compress_zlib(), con_read(), con_read_line(), contact_alloc(), contact_detail_alloc(), contact_name(), decompress_block_lzo(), decompress_bzip(), decompress_lzo(), decompress_zlib(), deprecated_hmac_digest(), deprecated_scramble_encrypt(), deprecated_scramble_import(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), deprecated_symmetric_vector(), dkim_signature_create(), dkim_signature_verify(), domain_alloc(), ed25519_private_get(), ed25519_private_set(), ed25519_public_get(), ed25519_public_set(), ed25519_sign(), ed25519_verify(), encrypted_chunk_get(), encrypted_chunk_set(), ephemeral_chunk_set(), file_load(), file_read(), hash_digest(), hex_decode_st(), hex_encode_st(), hmac_digest(), http_body(), http_parse_origin(), imap_command_parser(), imap_fetch_bodystructure(), int16_conv_st(), int32_conv_st(), int64_conv_st(), int8_conv_st(), magma_folder_alloc(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_discover_insertion_point(), mail_extract_address(), mail_get_boundary(), mail_insert_chunk_base64(), mail_mime_header(), mail_mime_part(), mail_mod_subject(), mail_modify_part(), mail_store_message_data(), message_alloc(), meta_data_fetch_shard(), meta_data_insert_shard(), meta_folder_stats_tag_alloc(), naked_message_get(), naked_message_set(), pl_data_get(), pop_num_parse(), pop_pass_parse(), pop_top_parse(), pop_user_parse(), prime_field_write(), prime_header_write(), prime_pem_unwrap(), prime_pem_wrap(), prime_reader_open(), prime_unpack(), qp_decode(), qp_encode(), rand_choices(), rand_write(), register_data_insert_user(), sanity_check(), scramble_encrypt(), scramble_import(), secp256k1_compute_kek(), secp256k1_private_get(), secp256k1_private_set(), secp256k1_public_get(), secp256k1_public_set(), sess_get(), signature_full_verify(), signature_tree_verify(), slots_get(), smtp_check_greylist(), smtp_data_finish(), smtp_data_read(), sql_insert_conn(), sql_num_rows_conn(), sql_query_conn(), sql_query_res_conn(), sql_write_conn(), st_append_opts(), st_append_out(), st_bitwise(), st_char_get(), st_copy_in(), st_dupe_opts(), st_empty_out(), st_empty_variadic(), st_import_opts(), st_merge_opts(), st_not(), st_nullify(), st_populated_variadic(), st_replace(), st_set(), st_uchar_get(), st_vaprint_opts(), st_vsprint(), st_wipe(), st_write_variadic(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_encrypt(), stacie_realm_cipher(), stacie_realm_tag(), stacie_realm_vector(), symmetric_decrypt(), symmetric_encrypt(), symmetric_vector(), tank_store(), tcp_status(), tok_get_count_st(), tok_get_st(), uint16_conv_st(), uint16_get_no(), uint16_put_no(), uint24_get_no(), uint24_put_no(), uint32_conv_st(), uint32_get_no(), uint32_put_no(), uint64_conv_st(), uint8_conv_st(), url_decode(), url_encode(), user_config_entry_alloc(), virus_check(), xml_encode(), zbase32_decode(), and zbase32_encode().

5.67.1.3 void st_data_set (stringer_t * s, void * data)

Set the underlying data of a jointed managed string.

Parameters:

s the managed string to be adjusted.

data the data buffer to be attached to the input managed string.

Note:

The underlying data of *s* will be released, unless it contains foreign data.

Returns:

This function does not return a value.

Definition at line 92 of file data.c.

References block_t, BLOCK_T, FOREIGNDATA, JOINTED, log_pedantic, managed_t, MANAGED_T, MAPPED_T, MEMORYBUF, mm_free(), mm_sec_free(), mm_sec_secured(), nuller_t, NULLER_T, placer_t, PLACER_T, SECURE, st_info_opts(), and st_valid_jointed().

Referenced by contact_name(), ephemeral_chunk_set(), smtp_data_finish(), smtp_data_read(), and st_dupe_opts().

5.67.1.4 bool_t st_empty_out (stringer_t * s, uchr_t ** ptr, size_t * len)

Determine whether the specified managed string is empty or not, and store its underlying data and data length.

Parameters:

s the input managed string.

ptr a pointer to a null-terminated string that will receive the address of the managed string's underlying data.

len a pointer to store the size of the managed string's underlying data buffer.

Returns:

true if string is NULL or uninitialized or empty; false otherwise.

Definition at line 76 of file data.c.

References st_data_get(), and st_length_get().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), auth_sanitise_address(), base64_decode(), base64_decode_mod(), base64_decode_opts(), base64_encode(), base64_encode_mod(), base64_encode_opts(), base64_encode_wrap(), chunk_buffer_size(), chunk_header_size(), chunk_header_type(), client_write(), contact_name(), contact_validate_detail(), contact_validate_name(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), ecies_decrypt(), ecies_encrypt(), ephemeral_chunk_set(), hex_count_st(), hex_decode_st(), hex_encode_st(), hex_valid_st(), http_parse_origin(), imap_command_log_safe(), imap_count_folder_levels(), imap_fetch_body_portion(), imap_valid_folder_name(), imap_valid_sequence(), lower_st(), mail_count_received(), mail_extract_address(), mail_mime_encode_part(), naked_message_get(), part_decrypt(), part_encrypt(), prime_field_write(), prime_header_read(), qp_decode(), qp_encode(), smtp_parse_auth(), st_cmp_ci_ends(), st_cmp_ci_eq(), st_cmp_ci_starts(), st_cmp_cs_ends(), st_cmp_cs_eq(), st_cmp_cs_starts(), st_replace(), st_search_chr(), st_search_ci(), st_search_cs(), st_swap(), st_write_variadic(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_encrypt(), stacie_realm_key(), symmetric_decrypt(), symmetric_encrypt(), upper_st(), url_decode(), url_encode(), url_valid_st(), utf8_length_st(), utf8_valid_st(), zbase32_decode(), and zbase32_encode().

5.67.1.5 bool_t st_empty_variadic (ssize_t len, ...)

A simple method for checking multiple managed strings to see if any are empty.

Parameters:

len the number of strings being passed in.

va_list a list of managed strings to check for emptiness

Returns:

true if any of the strings are NULL, uninitialized or empty; false if every string has at least one byte of data.

Definition at line 48 of file data.c.

References st_data_get(), and st_length_get().

5.67.1.6 `bool_t st_populated_variadic (ssize_t len, ...)`

A simple method for checking multiple managed strings to ensure all of the provided strings have contain at least one character of data.

Parameters:

- len* the number of strings being passed in.
- va_list* a list of managed strings to be validated.

Returns:

true if all of the strings are populated with at least one byte of data, otherwise false.

Definition at line 19 of file data.c.

References `st_data_get()`, and `st_length_get()`.

5.67.1.7 `stringer_t* st_set (stringer_t * s, uint8_t set, size_t len)`

Sets a block of memory to a specified value.

Parameters:

- s* the managed string to be overwritten with the value of set.
- set* the byte value to be write into the string.
- len* the number of bytes to set to the provided value.

Returns:

a pointer to the block of memory passed to the function.

Definition at line 246 of file data.c.

References `log_pedantic`, `MEMORYBUF`, `mm_set()`, `st_avail_get()`, `st_data_get()`, `st_info_opts()`, `st_length_set()`, `st_valid_opts()`, and `st_valid_tracked()`.

Referenced by `encrypted_chunk_get()`, `encrypted_chunk_set()`, and `slots_key()`.

5.67.1.8 `uchar_t* st_uchar_get (stringer_t * s)`

Retrieve an unsigned character pointer to a managed string's data buffer.

See also:

[st_data_get\(\)](#)

Parameters:

- s* the input managed string.

Returns:

NULL on failure or for an improperly constructed string; otherwise, a pointer to the string's data.

Definition at line 206 of file data.c.

References `st_data_get()`.

Referenced by `dkim_signature_create()`, `http_parse_origin()`, and `prime_pem_unwrap()`.

5.67.1.9 void st_wipe (stringer_t * s)

Wipe all of a managed string's allocated memory and if applicable, reset its length.

Parameters:

s the managed string to be wiped.

Returns:

This function returns no value.

Definition at line 216 of file data.c.

References log_pedantic, MEMORYBUF, mm_wipe(), st_avail_get(), st_data_get(), st_info_opts(), st_length_set(), st_valid_opts(), and st_valid_tracked().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), ed25519_private_get(), ed25519_public_get(), ip_standard(), org_key_get(), org_signet_get(), secp256k1_private_get(), user_key_get(), user_request_get(), and user_signet_get().

5.68 src/providers/storage/data.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t tank_insert_object` (`int64_t transaction`, `uint64_t hnum`, `uint64_t tnum`, `uint64_t unum`, `uint64_t size`, `uint64_t flags`)
Insert a tank object into the mysql database.
- `bool_t tank_delete_object` (`int64_t transaction`, `uint64_t hnum`, `uint64_t tnum`, `uint64_t unum`, `uint64_t onum`)
Delete a tank object from the mysql database.

5.68.1 Function Documentation

5.68.1.1 `bool_t tank_delete_object` (`int64_t transaction`, `uint64_t hnum`, `uint64_t tnum`, `uint64_t unum`, `uint64_t onum`)

Delete a tank object from the mysql database.

Parameters:

transaction a mysql connection identifier for which a transaction has been started.

hnum the host number.

tnum the storage tank number.

unum the usernum of the user that owns the object.

onum the stored object id.

Returns:

false on failure, or true on success.

Definition at line 67 of file data.c.

References `log_pedantic`, `mm_wipe()`, and `stmt_exec_affected_conn()`.

Referenced by `tank_delete()`.

5.68.1.2 `uint64_t tank_insert_object` (`int64_t transaction`, `uint64_t hnum`, `uint64_t tnum`, `uint64_t unum`, `uint64_t size`, `uint64_t flags`)

Insert a tank object into the mysql database.

Parameters:

transaction a mysql connection identifier for which a transaction has been started.

hnum the host number.

tnum the storage tank number.

unum the usernum of the user that owns the object.

size the length, in bytes, of the object to be stored.

flags 0 on failure, or the objectnum field for the newly inserted object.

Definition at line 19 of file data.c.

References `mm_wipe()`, and `stmt_insert_conn()`.

Referenced by `tank_store()`.

5.69 src/servers/http/data.c File Reference

```
#include "magma.h"
```

Functions

- void [http_data_free](#) ([http_data_t](#) *data)
Free an http data object.
- [http_data_t](#) * [http_data_get](#) ([connection_t](#) *con, [HTTP_DATA](#) source, [chr_t](#) *name)
Get a name/value pair associated with an http connection, by name.
- void [http_data_value_decode](#) ([stringer_t](#) *string)
Decode an escaped URI component into its original data.
- [int_t](#) [http_data_value_parse](#) ([connection_t](#) *con, [HTTP_DATA](#) source, [placer_t](#) pair)
Parse a string containing a name/value pair, and place it in the specified connection's pairs holder.
- [http_data_t](#) * [http_data_header_parse_line](#) ([chr_t](#) *buf, [size_t](#) len)
Parse a data buffer into a http header name/value pair.
- [http_data_t](#) * [http_data_header_parse](#) ([connection_t](#) *con)
Parse the current line of input from an http client connection into a http header name/value pair.

5.69.1 Function Documentation

5.69.1.1 void http_data_free (http_data_t * data)

Free an http data object. data.c

Parameters:

data the http data object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file data.c.

References [mm_free\(\)](#), [http_data_t::name](#), [st_cleanup](#), and [http_data_t::value](#).

Referenced by [http_data_value_parse\(\)](#), [http_parse_header\(\)](#), [http_parse_pairs\(\)](#), and [portal_upload\(\)](#).

5.69.1.2 http_data_t* http_data_get (connection_t * con, HTTP_DATA source, chr_t * name)

Get a name/value pair associated with an http connection, by name.

Note:

The name/value pairs searched can be supplied by a client through http request headers, or through GET or POST data.

Parameters:

con the connection object to be queried.

source the source of the client supplied value pair: HTTP_DATA_GET, HTTP_DATA_POST or HTTP_DATA_ANY.

name the name associated with the name/value pair to be retrieved.

Returns:

NULL on failure, or a pointer to the http name/value data pair requested on success.

TODO: We could just use the index find function.

Definition at line 34 of file data.c.

References data, HTTP_DATA_ANY, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), http_data_t::name, NULLER, http_data_t::source, and st_cmp_ci_eq().

Referenced by contact_business(), contact_print_form(), contact_process(), http_body(), http_response_allow_cross(), multipart_get_boundary(), register_business_step1(), register_business_step2(), register_process(), and teacher_process().

5.69.1.3 http_data_t* http_data_header_parse (connection_t * con)

Parse the current line of input from an http client connection into a http header name/value pair.

Parameters:

con the connection object of the http client to be read.

Returns:

NULL on failure, or a pointer to an http header name/value pair on success.

Definition at line 226 of file data.c.

References http_data_header_parse_line(), st_char_get(), and st_length_get().

Referenced by http_parse_header().

5.69.1.4 http_data_t* http_data_header_parse_line (chr_t * buf, size_t len)

Parse a data buffer into a http header name/value pair.

Parameters:

con the connection object of the http client to be read.

Returns:

NULL on failure, or a pointer to an http header name/value pair on success.

Definition at line 163 of file data.c.

References HTTP_DATA_HEADER, mm_alloc(), http_data_t::name, http_data_t::source, st_free(), st_import(), and http_data_t::value.

Referenced by http_data_header_parse(), and portal_upload().

5.69.1.5 void http_data_value_decode (stringer_t * string)

Decode an escaped URI component into its original data.

Note:

Since the un-escaped string will always be at least as small as the encoded value, the input managed string is transformed in place.

Parameters:

string a managed string containing the escaped URI data to be decoded.

Returns:

This function returns no value.

Definition at line 68 of file data.c.

References `hex_decode_chr()`, `length`, `st_char_get()`, `st_length_get()`, and `st_length_set()`.

Referenced by `http_data_value_parse()`.

5.69.1.6 int_t http_data_value_parse (connection_t * con, HTTP_DATA source, placer_t pair)

Parse a string containing a name/value pair, and place it in the specified connection's pairs holder.

Note:

Each name/value pair is assumed to be delimited with a '=' character, and each half is URI-decoded before storage.

Parameters:

con a pointer to the connection object of the http client submitting the user data to be parsed.

source an HTTP_DATA value specifying the source of the name/value pair (can be HTTP_DATA_HEADER, HTTP_DATA_GET, or HTTP_DATA_POST).

pair a placer pointing to a string containing the name/value pair to be parsed.

Returns:

0 on general or parsing failure, or 1 if the specified input buffer was successfully parsed and stored.

Definition at line 111 of file data.c.

References `CONTIGUOUS`, `data`, `HEAP`, `magma_t::http`, `http_data_free()`, `http_data_value_decode()`, `inx_insert()`, `magma_t::log`, `log_pedantic`, `M_TYPE_STRINGER`, `magma`, `MANAGED_T`, `mm_alloc()`, `http_data_t::name`, `pl_empty()`, `placer_t`, `http_data_t::source`, `multi_t::st`, `st_char_get()`, `st_dupe_opts()`, `st_free()`, `st_length_int()`, `tok_get_pl()`, `multi_t::val`, and `http_data_t::value`.

Referenced by `http_parse_pairs()`.

5.70 src/core/strings/info.c File Reference

```
#include "magma.h"
```

Functions

- `const chr_t * st_info_type (uint32_t opts)`
Get a readable description of a managed string's data type.
- `const chr_t * st_info_layout (uint32_t opts)`
Get a readable description of a managed string's data layout.
- `const chr_t * st_info_allocator (uint32_t opts)`
Get a readable description of a managed string's allocator type.
- `chr_t * st_info_opts (uint32_t opts, chr_t *s, size_t len)`
Get a readable description of all of a managed string's option flags.

Variables

- `chr_t * st_option_flags []`
- `chr_t * st_option_types []`
- `chr_t * st_option_layouts []`
- `chr_t * st_option_allocators []`

5.70.1 Function Documentation

5.70.1.1 `const chr_t* st_info_allocator (uint32_t opts)`

Get a readable description of a managed string's allocator type.

Parameters:

opts a value containing the managed string's option mask.

Returns:

a pointer to a null-terminated string containing a description of the managed string's allocator type.

Definition at line 96 of file info.c.

References `HEAP`, `SECURE`, `st_option_allocators`, and `STACK`.

Referenced by `st_info_opts()`.

5.70.1.2 `const chr_t* st_info_layout (uint32_t opts)`

Get a readable description of a managed string's data layout.

Parameters:

opts a value containing the managed string's option mask.

Returns:

a pointer to a null-terminated string containing a description of the managed string's data layout.

Definition at line 75 of file info.c.

References CONTIGUOUS, JOINTED, and st_option_layouts.

Referenced by st_info_opts().

5.70.1.3 chr_t* st_info_opts (uint32_t opts, chr_t * s, size_t len)

Get a readable description of all of a managed string's option flags.

Parameters:

opts a value containing the managed string's option mask.

s a pointer to a null-terminated string to receive the options description.

len the length, in bytes, of the description output buffer.

Returns:

NULL on failure, or a pointer to the output buffer containing the managed string's options description on success.

LOW: Turn this into a loop.

Definition at line 122 of file info.c.

References FOREIGNDATA, NULLER, st_append, st_char_get(), st_cleanup, st_empty, st_info_allocator(), st_info_layout(), st_info_type(), st_length_int(), and st_option_flags.

Referenced by st_alloc_opts(), st_avail_get(), st_avail_set(), st_data_get(), st_data_set(), st_dupe_opts(), st_free(), st_import_opts(), st_length_get(), st_length_set(), st_merge_opts(), st_opt_set(), st_opt_test(), st_realloc(), st_set(), st_vaprint_opts(), st_vsprint(), and st_wipe().

5.70.1.4 const chr_t* st_info_type (uint32_t opts)

Get a readable description of a managed string's data type.

Parameters:

opts a value containing the managed string's option mask.

Returns:

a pointer to a null-terminated string containing a description of the managed string's data type.

Definition at line 42 of file info.c.

References BLOCK_T, CONSTANT_T, MANAGED_T, MAPPED_T, NULLER_T, PLACER_T, and st_option_types.

Referenced by st_info_opts().

5.70.2 Variable Documentation**5.70.2.1 chr_t* st_option_allocators[]****Initial value:**

```
{
    "UNKNOWN",
```

```
"STACK",  
"HEAP",  
"SECURE"  
}
```

Definition at line 30 of file info.c.

Referenced by st_info_allocator().

5.70.2.2 chr_t* st_option_flags[]

Initial value:

```
{  
    "FOREIGNDATA"  
}
```

Definition at line 10 of file info.c.

Referenced by st_info_opts().

5.70.2.3 chr_t* st_option_layouts[]

Initial value:

```
{  
    "UNKNOWN",  
    "CONTIGUOUS",  
    "JOINTED"  
}
```

Definition at line 24 of file info.c.

Referenced by st_info_layout().

5.70.2.4 chr_t* st_option_types[]

Initial value:

```
{  
    "UNKNOWN",  
    "CONSTANT",  
    "PLACER",  
    "NULLER",  
    "BLOCK",  
    "MANAGED",  
    "MAPPED"  
}
```

Definition at line 14 of file info.c.

Referenced by st_info_type().

5.71 src/core/strings/length.c File Reference

```
#include "magma.h"
```

Functions

- `size_t st_length_get (stringer_t *s)`
Return the length of the data in a managed string.
- `int_t st_length_int (stringer_t *s)`
Length.
- `size_t st_length_set (stringer_t *s, size_t len)`
Set the data length for a managed string that supports data length tracking.
- `size_t st_avail_get (stringer_t *s)`
Return the total data buffer size allocated for a managed string.
- `size_t st_avail_set (stringer_t *s, size_t avail)`
Set the total data buffer size allocated for a managed string.

5.71.1 Function Documentation

5.71.1.1 size_t st_avail_get (stringer_t * s)

Return the total data buffer size allocated for a managed string.

Parameters:

`s` the input managed string.

Returns:

0 on failure, or the total buffer size in bytes on success.

Definition at line 146 of file length.c.

References `log_pedantic`, `managed_t`, `MANAGED_T`, `mapped_t`, `MAPPED_T`, `MEMORYBUF`, `st_info_opts()`, `st_length_get()`, and `st_valid_opts()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `base64_decode()`, `base64_decode_mod()`, `base64_encode()`, `base64_encode_mod()`, `base64_encode_wrap()`, `client_read()`, `client_read_line()`, `con_read()`, `con_read_line()`, `deprecated_hmac_digest()`, `ed25519_private_get()`, `ed25519_public_get()`, `ed25519_sign()`, `file_load()`, `file_read()`, `hash_digest()`, `hex_decode_st()`, `hex_encode_st()`, `hmac_digest()`, `host_platform()`, `host_version()`, `imap_parse_address_put()`, `ip_presentation()`, `ip_reversed()`, `ip_standard()`, `ip_subnet()`, `mail_add_required_headers()`, `meta_data_fetch_shard()`, `org_key_get()`, `org_signet_get()`, `rand_choices()`, `rand_write()`, `sanity_check()`, `secp256k1_compute_kek()`, `secp256k1_private_get()`, `secp256k1_public_get()`, `serialize_int16()`, `serialize_int32()`, `serialize_int64()`, `serialize_ns()`, `serialize_st()`, `serialize_uint16()`, `serialize_uint32()`, `serialize_uint64()`, `st_append_opts()`, `st_append_out()`, `st_bitwise()`, `st_copy_in()`, `st_dupe_opts()`, `st_not()`, `st_output()`, `st_set()`, `st_trim()`, `st_vsprint()`, `st_wipe()`, `st_write_variadic()`, `stacie_create_nonce()`, `stacie_create_salt()`, `stacie_create_shard()`, `time_print_gmt()`, `time_print_local()`, `user_key_get()`, `user_request_get()`, and `user_signet_get()`.

5.71.1.2 size_t st_avail_set (stringer_t * s, size_t avail)

Set the total data buffer size allocated for a managed string.

Parameters:

- s* the input managed string.
- avail* the new buffer size for the managed string.

Returns:

0 on failure, or the managed string's new buffer size in bytes on success.

Definition at line 183 of file length.c.

References log_pedantic, managed_t, MANAGED_T, mapped_t, MAPPED_T, MEMORYBUF, st_info_opts(), and st_valid_avail().

5.71.1.3 size_t st_length_get (stringer_t * s)

Return the length of the data in a managed string.

Parameters:

- s* the input managed string.

Returns:

0 on failure, or the length of the managed string's data in bytes.

Definition at line 15 of file length.c.

References block_t, BLOCK_T, constant_t, CONSTANT_T, data, log_pedantic, managed_t, MANAGED_T, mapped_t, MAPPED_T, MEMORYBUF, nuller_t, NULLER_T, placer_t, PLACER_T, st_info_opts(), and st_valid_opts().

Referenced by _serialize_ec_pubkey(), aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), aes_cipher_key(), aes_tag_shard(), aes_vector_shard(), alert_alloc(), alias_alloc(), auth_challenge(), auth_data_fetch(), auth_data_update_legacy(), auth_login(), auth_response(), auth_sanitize_address(), auth_sanitize_username(), base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), cache_add(), cache_append(), cache_config(), cache_decrement(), cache_delete(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_set_value(), cache_silent_add(), chunk_buffer_read(), chunk_header_read(), client_read(), client_read_line(), compress_bzip(), compress_import(), compress_lzo(), compress_zlib(), con_read(), con_read_line(), con_write_st(), config_value_set(), contact_alloc(), contact_business_valid_email(), contact_detail_alloc(), contact_detail_delete(), contact_detail_upsert(), contact_folder_create(), contact_folder_rename(), contact_insert(), contact_print_form(), contact_print_message(), contact_update(), decompress_block_lzo(), deprecated_hmac_digest(), deprecated_scramble_decrypt(), deprecated_scramble_encrypt(), deprecated_scramble_import(), deprecated_symmetric_key(), deserialize_int16(), deserialize_int32(), deserialize_int64(), deserialize_ns(), deserialize_st(), deserialize_uint16(), deserialize_uint32(), deserialize_uint64(), dkim_signature_create(), dkim_signature_verify(), domain_alloc(), double_conv(), dspam_train(), ed25519_private_set(), ed25519_public_set(), ed25519_sign(), ed25519_verify(), encrypted_chunk_get(), encrypted_chunk_set(), ephemeral_chunk_get(), ephemeral_chunk_set(), float_conv(), get_header_value_noopt(), hash_digest(), hex_decode_opts(), hex_decode_st(), hex_encode_opts(), hex_encode_st(), hex_encode_st_debug(), hmac_digest(), http_body(), http_data_header_parse(), http_data_value_decode(), http_load_file(), http_page_get(), http_parse_context(), http_parse_header(), http_parse_pairs(), http_response(), imap_append_message(), imap_build_array(), imap_command_parser(), imap_copy(), imap_fetch_body(), imap_fetch_body_header(), imap_fetch_bodystructure(), imap_fetch_envelope(), imap_fetch_message(), imap_fetch_parse_partial(), imap_folder_compare(), imap_id(), imap_init(), imap_narrow_folders(), imap_parse_address_breaker(), imap_parse_address_part(), imap_parse_address_put(), imap_process(), imap_search_messages_date(), imap_search_messages_header(), imap_starttls(), imap_status(), imap_valid_folder_name(), int16_conv_st(), int32_conv_st(), int64_conv_st(), int8_conv_st(), ip_subnet(), line_pl_st(), magma_folder_alloc(), magma_folder_insert(), magma_folder_rename(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_add_required_headers(), mail_build_signature(), mail_db_insert_duplicate_message(), mail_db_insert_message(), mail_discover_encoding(), mail_discover_insertion_point(), mail_discover_type(), mail_get_boundary(), mail_get_chunk(), mail_header_end(), mail_header_fetch_all(), mail_header_fetch_cleaned(), mail_header_pop(), mail_headers(), mail_insert_chunk_base64(), mail_insert_chunk_text(), mail_load_message_top(), mail_message(), mail_message_cleanup(), mail_mime_boundary(), mail_mime_child(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_count(), mail_mime_encoding(), mail_mime_header(), mail_mime_part(), mail_mime_type(), mail_mime_type_group(), mail_mime_type_parameters_key(), mail_mime_type_parameters_value(), mail_mime_type_sub(), mail_mime_update(), mail_mod_subject(), mail_modify_part(), mail_signature_add(), mail_store_message_data(), message_alloc(), meta_data_delete_tag(), meta_data_fetch_shard(), meta_data_insert_folder(), meta_data_insert_keys(), meta_data_insert_shard(), meta_data_

insert_tag(), meta_data_update_folder_name(), meta_folder_stats_tag_alloc(), meta_update_realms(), mt_get_length(), naked_message_get(), naked_message_set(), nvp_parse(), org_signet_fingerprint(), org_signet_set(), org_signet_verify(), pl_length_get(), pop_num_parse(), pop_pass_parse(), pop_retr(), pop_top(), pop_top_parse(), pop_user_parse(), portal_endpoint_alert_acknowledge(), portal_endpoint_attachments_add(), portal_endpoint_attachments_progress(), portal_endpoint_attachments_remove(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_edit(), portal_endpoint_contacts_list(), portal_endpoint_contacts_load(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), portal_endpoint_folders_add(), portal_endpoint_folders_list(), portal_endpoint_folders_remove(), portal_endpoint_folders_rename(), portal_endpoint_folders_tags(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_load(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_endpoint_search(), portal_message_attachments(), portal_print_login(), portal_settings_changepass(), portal_upload(), portal_validate_request(), prime_key_decrypt(), prime_pem_unwrap(), prime_pem_wrap(), prime_reader_open(), prime_unpack(), qp_decode(), rand_write(), register_abuse_check_blocklist(), register_business_validate_password(), register_business_validate_username(), register_captcha_generate(), register_data_check_username(), register_data_insert_user(), register_print_captcha(), register_print_message(), register_print_step1(), register_print_step2(), register_print_step3(), relay_config(), relay_set_value(), scramble_decrypt(), scramble_encrypt(), scramble_import(), secp256k1_private_set(), secp256k1_public_set(), serialize_int16(), serialize_int32(), serialize_int64(), serialize_ns(), serialize_st(), serialize_uint16(), serialize_uint32(), serialize_uint64(), servers_config(), servers_set_value(), signature_full_get(), signature_full_verify(), signature_tree_add(), signature_tree_get(), signature_tree_verify(), slots_get(), slots_key(), slots_set(), smtp_accept_message(), smtp_auth_plain(), smtp_check_authorized_from(), smtp_check_filters(), smtp_check_greylist(), smtp_check_receive_quota(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), smtp_client_send_rcptto(), smtp_data_outbound(), smtp_fetch_authorization(), smtp_fetch_inbound(), smtp_forward_message(), smtp_insert_spamsig(), smtp_update_receive_stats(), sql_insert_conn(), sql_num_rows_conn(), sql_query_conn(), sql_query_res_conn(), sql_write_conn(), st_append_opts(), st_append_out(), st_avail_get(), st_bitwise(), st_dupe_opts(), st_empty_out(), st_empty_variadic(), st_length_int(), st_merge_opts(), st_not(), st_populated_variadic(), st_realloc(), st_trim(), stacie_decrypt(), stacie_encrypt(), stacie_realm_cipher(), stacie_realm_tag(), stacie_realm_vector(), statistics_process(), symmetric_key(), tank_store(), teacher_print_form(), teacher_print_message(), teacher_process(), tok_get_count_st(), tok_get_st(), tok_pop_init_st(), uint16_conv_st(), uint16_get_no(), uint24_get_no(), uint32_conv_st(), uint32_get_no(), uint64_conv_st(), uint8_conv_st(), url_decode(), user_config_delete(), user_config_entry_alloc(), user_config_upsert(), user_request_generate(), user_request_rotation(), user_request_set(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_set(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), user_signet_verify_self(), and virus_check().

5.71.1.4 int_t st_length_int (stringer_t * s)

Length. Return the length of a managed string as an integer.

Parameters:

s the managed string as type stringer_t *.

Returns:

the string length as an integer, capped at INT_MAX.

Definition at line 70 of file length.c.

References log_pedantic, st_length_get(), and st_valid_opts().

Referenced by api_endpoint_auth(), auth_data_fetch(), auth_login(), auth_response(), cache_add(), cache_append(), cache_config(), cache_decrement(), cache_delete(), cache_get(), cache_get_u64(), cache_increment(), cache_output_settings(), cache_set(), cache_set_u64(), cipher_name(), client_read(), client_read_line(), client_write(), config_load_cmdline_settings(), config_load_database_settings(), config_load_file_settings(), config_output_value(), config_output_value_generic(), contact_abuse_checks(), contact_abuse_increment_history(), contact_detail_delete(), contact_detail_upsert(), digest_name(), dkim_start(), dmtpl_helo(), dmtpl_init(), double_conv(), float_conv(), http_body(), http_data_value_parse(), http_load_file(), http_parse_header(), http_parse_method(), http_print_301(), http_response_allow_cross(), http_response_cookie(), http_response_header(), http_response_options(), imap_append(), imap_append_message(), imap_capability(), imap_check(), imap_close(), imap_command_log_safe(), imap_copy(), imap_create(), imap_delete(), imap_examine(), imap_expunge(), imap_fetch(), imap_id(), imap_idle(), imap_init(), imap_invalid(), imap_list(), imap_login(), imap_logout(), imap_lsub(), imap_narrow_messages(), imap_noop(), imap_process(), imap_rename(), imap_search(), imap_select(), imap_status(), imap_store(), imap_subscribe(), imap_unsubscribe(), lock_get(), lock_release(), mail_create_directory(), mail_load_message(), mail_message_path(), mail_signature_add(), mail_store_message(), meta_crypto_keys_create(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_

data_fetch_shard(), meta_data_insert_keys(), meta_data_insert_shard(), meta_update_keys(), meta_update_realms(), pl_length_int(), pop_pass(), portal_config_entry(), portal_endpoint_auth(), prime_start(), process_kill(), protocol_secure(), register_business_step1(), register_data_insert_user(), register_session_cache(), register_session_get(), relay_config(), relay_output_settings(), serial_get(), serial_increment(), serial_reset(), servers_config(), servers_output_settings(), servers_set_value(), sess_get(), smtp_auth_login(), smtp_auth_plain(), smtp_bounce(), smtp_check_filters(), smtp_check_greylist(), smtp_check_rbl(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_mailfrom(), smtp_client_send_nullfrom(), smtp_client_send_rcptto(), smtp_ehlo(), smtp_fetch_authorization(), smtp_fetch_inbound(), smtp_helo(), smtp_init(), smtp_rcpt_to(), smtp_reply(), spf_check(), spool_check(), spool_cleanup(), spool_mktemp(), spool_start(), spool_stop(), sql_query_conn(), st_info_opts(), stacie_derive_key(), stacie_derive_token(), user_config_delete(), user_config_upsert(), and virus_start().

5.71.1.5 size_t st_length_set (stringer_t * s, size_t len)

Set the data length for a managed string that supports data length tracking.

Parameters:

- s* the input managed string.
- len* the new value of the managed string's data length.

Returns:

- 0 on error, or the new data length on success.

Definition at line 99 of file length.c.

References log_pedantic, managed_t, MANAGED_T, mapped_t, MAPPED_T, MEMORYBUF, placer_t, PLACER_T, st_info_opts(), and st_valid_tracked().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), auth_sanitise_address(), auth_sanitise_username(), base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), chunk_header_write(), client_read(), client_read_line(), con_read(), con_read_line(), contact_name(), decompress_block_lzo(), decompress_bzip(), decompress_lzo(), decompress_zlib(), deprecated_hmac_digest(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), deprecated_symmetric_key(), deprecated_symmetric_vector(), deserialize_st(), ed25519_private_get(), ed25519_public_get(), ed25519_sign(), encrypted_chunk_set(), ephemeral_chunk_set(), file_load(), file_read(), hash_digest(), hex_decode_st(), hex_encode_st(), hex_encode_st_debug(), hmac_digest(), http_data_value_decode(), http_load_file(), http_parse_header(), imap_build_array(), imap_parse_address_part(), imap_parse_address_put(), imap_parse_literal(), imap_parse_qstring(), imap_starttls(), imap_valid_folder_name(), ip_presentation(), ip_reversed(), ip_standard(), mail_add_required_headers(), mail_extract_address(), mail_extract_tag(), mail_header_fetch_cleaned(), mail_load_message(), mail_load_message_top(), mail_message_cleanup(), mail_mime_generate_boundary(), pop_starttls(), prime_field_write(), prime_header_write(), qp_decode(), rand_choices(), rand_write(), secp256k1_compute_kek(), secp256k1_private_get(), secp256k1_public_get(), serialize_int16(), serialize_int32(), serialize_int64(), serialize_ns(), serialize_st(), serialize_uint16(), serialize_uint32(), serialize_uint64(), smtp_data_finish(), smtp_data_read(), smtp_starttls(), st_append_opts(), st_append_out(), st_bitwise(), st_copy_in(), st_dupe_opts(), st_import_opts(), st_merge_opts(), st_not(), st_nullify(), st_replace(), st_set(), st_trim(), st_vaprint_opts(), st_vsprint(), st_wipe(), st_write_variadic(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_encrypt(), symmetric_decrypt(), symmetric_encrypt(), symmetric_key(), symmetric_vector(), time_print_gmt(), time_print_local(), uint16_put_no(), uint24_put_no(), uint32_put_no(), url_decode(), url_encode(), zbase32_decode(), and zbase32_encode().

5.72 src/core/strings/multi.c File Reference

```
#include "magma.h"
```

Functions

- [multi_t mt_get_null](#) (void)
Return an empty multi-type object.
- [bool_t mt_is_empty](#) (multi_t multi)
Determine whether a multi-type object is empty.
- [bool_t mt_is_number](#) (multi_t multi)
Determine whether a multi-type object is a number.
- [uint64_t mt_get_number](#) (multi_t multi)
Get the value of a numerical multi-type object.
- [char * mt_get_char](#) (const multi_t *multi)
Get a character pointer to the value of a multi-type object.
- [size_t mt_get_length](#) (multi_t multi)
Get the length of the data associated with a multi-type object.
- [M_TYPE mt_get_type](#) (multi_t multi)
Get the data type associated with a multi-type object.
- [multi_t mt_set_type](#) (multi_t multi, M_TYPE target)
Set the data type of a multi-type data object.
- [void mt_free](#) (multi_t multi)
Free the data underlying a multi-type data object (for managed and null-terminated strings).
- [multi_t mt_dupe](#) (multi_t multi)
Duplicate a multi-type object.
- [bool_t ident_mt_mt](#) (multi_t one, multi_t two)
Check to see if the values of two multi-type objects are identical.
- [int32_t cmp_mt_mt](#) (multi_t one, multi_t two)
Compare the values of two multi-type objects of the same data type.

5.72.1 Function Documentation

5.72.1.1 int32_t cmp_mt_mt (multi_t one, multi_t two)

Compare the values of two multi-type objects of the same data type. [multi.c](#)

Note:

The types of the two input objects must match, or be a comparison between a managed string and null-terminated string.

Parameters:

- one* the first multi-type object to be compared.
- two* the second multi-type object to be compared.

Returns:

Definition at line 682 of file multi.c.

References multi_t::binary, multi_t::dbl, multi_t::fl, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, log_options, log_pedantic, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, mt_get_type(), multi_t::ns, NULLER, multi_t::st, st_cmp_cs_eq(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by tree_cmp().

5.72.1.2 bool_t ident_mt_mt (multi_t one, multi_t two)

Check to see if the values of two multi-type objects are identical.

Note:

The types of the two input objects must match, or be a comparison between a managed string and null-terminated string.

Parameters:

- one* the first multi-type object to be compared.
- two* the second multi-type object to be compared.

Returns:

true if the two objects have identical values; false if they don't, or on failure.

Definition at line 567 of file multi.c.

References multi_t::binary, multi_t::dbl, multi_t::fl, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, log_options, log_pedantic, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, mt_get_type(), multi_t::ns, ns_length_get(), PLACER, multi_t::st, st_cmp_cs_eq(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by hashed_bucket_find_key(), hashed_delete(), linked_delete(), and linked_find().

5.72.1.3 multi_t mt_dupe (multi_t multi)

Duplicate a multi-type object.

Parameters:

- multi* the multi-type object to be examined.

Returns:

a copy of the input object (a deep copy for managed strings and null-terminated strings).

Definition at line 484 of file multi.c.

References `multi_t::binary`, `CONTIGUOUS`, `multi_t::dbl`, `multi_t::fl`, `HEAP`, `multi_t::i16`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `log_pedantic`, `M_TYPE_BOOLEAN`, `M_TYPE_DOUBLE`, `M_TYPE_FLOAT`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `MANAGED_T`, `mt_get_null()`, `multi_t::ns`, `ns_dupe()`, `multi_t::st`, `st_dupe_opts()`, `type()`, `multi_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, and `multi_t::val`.

Referenced by `hashed_bucket_alloc()`, `linked_record_alloc()`, and `tree_insert()`.

5.72.1.4 `void mt_free (multi_t multi)`

Free the data underlying a multi-type data object (for managed and null-terminated strings).

Parameters:

multi the multi-type object to be freed.

Returns:

This function returns no value.

Definition at line 462 of file `multi.c`.

References `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `multi_t::ns`, `ns_free()`, `multi_t::st`, `st_free()`, `multi_t::type`, and `multi_t::val`.

Referenced by `hashed_delete()`, `hashed_free()`, `hashed_truncate()`, `linked_record_free()`, `tree_delete()`, `tree_insert()`, and `tree_truncate()`.

5.72.1.5 `char* mt_get_char (const multi_t * multi)`

Get a character pointer to the value of a multi-type object.

Parameters:

multi a pointer to the multi-type object to be examined.

a pointer to the value of the input object, or `NULL` on failure.

Returns:

null on error

Definition at line 172 of file `multi.c`.

References `multi_t::binary`, `multi_t::bl`, `multi_t::dbl`, `multi_t::fl`, `multi_t::i16`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `log_options`, `M_LOG_INFO`, `M_LOG_STACK_TRACE`, `M_TYPE_BLOCK`, `M_TYPE_BOOLEAN`, `M_TYPE_DOUBLE`, `M_TYPE_FLOAT`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `multi_t::ns`, `multi_t::st`, `st_char_get()`, `type()`, `multi_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, and `multi_t::val`.

Referenced by `hashed_bucket()`.

5.72.1.6 `size_t mt_get_length (multi_t multi)`

Get the length of the data associated with a multi-type object.

Parameters:

multi the multi-type object to be examined.

Returns:

the length in bytes of the data associated with the input object, or 0 on failure.

Definition at line 249 of file multi.c.

References `log_options`, `M_LOG_PEDANTIC`, `M_LOG_STACK_TRACE`, `M_TYPE_BOOLEAN`, `M_TYPE_DOUBLE`, `M_TYPE_FLOAT`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `multi_t::ns`, `ns_length_get()`, `multi_t::st`, `st_length_get()`, `type()`, `multi_t::type`, and `multi_t::val`.

Referenced by `hashed_bucket()`.

5.72.1.7 `multi_t mt_get_null (void)`

Return an empty multi-type object.

Returns:

an empty multi-type object.

Definition at line 17 of file multi.c.

References `EMPTY`, and `multi_t::type`.

Referenced by `hashed_cursor_key_active()`, `hashed_cursor_key_next()`, `inx_cursor_key_active()`, `inx_cursor_key_next()`, `linked_cursor_key_active()`, `linked_cursor_key_next()`, `linked_record_get_key()`, `mt_dupe()`, `tree_cursor_key_next()`, and `tree_cursor_next()`.

5.72.1.8 `uint64_t mt_get_number (multi_t multi)`

Get the value of a numerical multi-type object.

Parameters:

multi the multi-type object to be examined.

Returns:

the numerical value of the input object, or 0 on failure.

Definition at line 116 of file multi.c.

References `multi_t::dbl`, `multi_t::fl`, `multi_t::i16`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `log_options`, `M_LOG_PEDANTIC`, `M_LOG_STACK_TRACE`, `M_TYPE_BLOCK`, `M_TYPE_BOOLEAN`, `M_TYPE_DOUBLE`, `M_TYPE_EMPTY`, `M_TYPE_FLOAT`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `type()`, `multi_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, and `multi_t::val`.

Referenced by `hashed_bucket()`.

5.72.1.9 `M_TYPE mt_get_type (multi_t multi)`

Get the data type associated with a multi-type object.

Parameters:

multi the multi-type object to be examined.

Returns:

the data type of the input object, or `EMPTY` on failure.

Definition at line 318 of file multi.c.

References EMPTY, log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_PLACER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, type(), and multi_t::type.

Referenced by cmp_mt_mt(), and ident_mt_mt().

5.72.1.10 bool_t mt_is_empty (multi_t *multi*)

Determine whether a multi-type object is empty.

Note:

All numeric (including boolean and floating point) types will always return false.

Parameters:

multi the multi-type object to be examined.

Returns:

true if the object is empty (by type or value), or false otherwise.

Definition at line 31 of file multi.c.

References multi_t::bl, log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_EMPTY, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, multi_t::ns, ns_empty(), multi_t::st, st_empty, type(), multi_t::type, and multi_t::val.

Referenced by config_load_cmdline_settings(), config_load_file_settings(), and tree_insert().

5.72.1.11 bool_t mt_is_number (multi_t *multi*)

Determine whether a multi-type object is a number.

Parameters:

multi the multi-type object to be examined. return true if the input object is any integer, boolean, or floating point; false otherwise.

Definition at line 80 of file multi.c.

References log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_EMPTY, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, type(), and multi_t::type.

Referenced by hashed_bucket().

5.72.1.12 multi_t mt_set_type (multi_t *multi*, M_TYPE *target*)

Set the data type of a multi-type data object.

Parameters:

multi the multi-type object to be adjusted.

target the value of the new data type for the input object.

Returns:

a copy of the modified multi-type data object.

Definition at line 391 of file multi.c.

References EMPTY, log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_PLACER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, type(), and multi_t::type.

Referenced by nvp_parse().

5.73 src/core/strings/nuller.c File Reference

```
#include "magma.h"
```

Functions

- `size_t ns_length_get (const chr_t *s)`
Return the length of a null-terminated string.
- `bool_t ns_empty (chr_t *s)`
Return whether a null-terminated string is empty or not.
- `bool_t ns_empty_out (chr_t *s, chr_t **ptr, size_t *len)`
Return whether a null-terminated string is empty or not, while storing its address and length.
- `int ns_length_int (chr_t *s)`
Return the length of a null-terminated string as an int, capped at INT_MAX.
- `chr_t * ns_alloc (size_t len)`
Allocate (and wipe) a buffer for a null-terminated string.
- `void ns_free (chr_t *s)`
Free a null-terminated string.
- `bool_t ns_populated_variadic (ssize_t len,...)`
A simple method for checking multiple NULL terminated strings to ensure all of the provided pointers lead to a buffer with at least one non-NULL character.
- `void ns_cleanup_variadic (ssize_t len,...)`
A checked cleanup function which can be used free a variable number of null-terminated strings.
- `chr_t * ns_dupe (chr_t *s)`
Duplicate a null-terminated string.
- `chr_t * ns_import (void *block, size_t len)`
Get a block of memory as a null-terminated string.
- `chr_t * ns_append (chr_t *s, chr_t *append)`
Append one string to another and return the result.
- `void ns_wipe (chr_t *s, size_t len)`
Zero out a null-terminated string.

5.73.1 Function Documentation

5.73.1.1 chr_t* ns_alloc (size_t len)

Allocate (and wipe) a buffer for a null-terminated string. [nuller.c](#)

Parameters:

len the length of the buffer to be allocated.

Returns:

NULL on failure or if len was 0; a pointer to the newly allocated memory otherwise.

Definition at line 68 of file nuller.c.

References `log_pedantic`, `mm_alloc()`, and `ns_wipe()`.

Referenced by `deserialize_ns()`, `mail_message_path()`, and `ns_append()`.

5.73.1.2 `chr_t* ns_append (chr_t * s, chr_t * append)`

Append one string to another and return the result.

Parameters:

s a pointer to a leading null-terminated string to to which the other string will be appended.

append a pointer to a null-terminated string to be appended to the leading string.

Returns:

NULL if a memory allocation failure occurred, or a pointer to the resulting string on success.

Definition at line 211 of file nuller.c.

References `log_pedantic`, `mm_copy()`, `ns_alloc()`, `ns_dupe()`, `ns_free()`, and `ns_length_get()`.

Referenced by `api_response()`, `portal_endpoint_error()`, and `portal_endpoint_response()`.

5.73.1.3 `void ns_cleanup_variadic (ssize_t len, ...)`

A checked cleanup function which can be used free a variable number of null-terminated strings.

See also:

[ns_free\(\)](#)

Parameters:

va_list a list of null-terminated strings to be freed.

Returns:

This function returns no value.

Definition at line 146 of file nuller.c.

References `ns_free()`.

5.73.1.4 `chr_t* ns_dupe (chr_t * s)`

Duplicate a null-terminated string.

Parameters:

s the null-terminated string to be duplicated.

Returns:

NULL on failure, or a pointer to a copy of the input string.

Definition at line 167 of file nuller.c.

References `length`, `log_info`, `log_pedantic`, `mm_alloc()`, `mm_copy()`, and `ns_length_get()`.

Referenced by `cache_set_value()`, `config_value_set()`, `mt_dupe()`, `ns_append()`, `relay_set_value()`, `servers_set_value()`, and `xml_get_xpath_ns()`.

5.73.1.5 `bool_t ns_empty (chr_t * s)`

Return whether a null-terminated string is empty or not.

Parameters:

s the input as a null-terminated string.

Returns:

true if string is NULL or zero length; false otherwise.

Definition at line 29 of file nuller.c.

References `ns_length_get()`.

Referenced by `cache_free()`, `cache_output_settings()`, `cache_set_value()`, `cache_validate()`, `config_output_value()`, `config_output_value_generic()`, `config_value_set()`, `dkim_signature_create()`, `lib_load()`, `mail_mime_get_media_type()`, `mt_is_empty()`, `relay_free()`, `relay_output_settings()`, `relay_set_value()`, `relay_validate()`, `servers_free()`, `servers_output_settings()`, `servers_set_value()`, `servers_validate()`, `sql_start()`, and `st_merge_opts()`.

5.73.1.6 `bool_t ns_empty_out (chr_t * s, chr_t ** ptr, size_t * len)`

Return whether a null-terminated string is empty or not, while storing its address and length.

Parameters:

s the input as a null-terminated string.

ptr a pointer address to receive a copy of the string's location.

len a pointer to a variable to receive the length of the string, in bytes.

Returns:

true if string is NULL or zero length; false otherwise.

Definition at line 41 of file nuller.c.

References `ns_length_get()`.

5.73.1.7 `void ns_free (chr_t * s)`

Free a null-terminated string.

Parameters:

s the string to be freed.

Returns:

This function returns no value.

Definition at line 95 of file nuller.c.

References `log_pedantic`, and `mm_free()`.

Referenced by `api_response()`, `cache_free()`, `cache_set_value()`, `config_free()`, `config_value_set()`, `ip_subnet_st()`, `mail_copy_message()`, `mail_load_message()`, `mail_message_path()`, `mail_remove_message()`, `mail_store_message()`, `mail_store_message_data()`, `mt_free()`, `ns_append()`, `ns_cleanup_variadic()`, `portal_endpoint_error()`, `portal_endpoint_response()`, `relay_free()`, `relay_set_value()`, `servers_free()`, and `servers_set_value()`.

5.73.1.8 `chr_t* ns_import (void *block, size_t len)`

Get a block of memory as a null-terminated string.

Parameters:

block a pointer to the buffer containing the data to be copied.

len the length, in bytes, of the data buffer to be copied.

Returns:

NULL on failure, or a pointer to the newly allocated null-terminated string on success.

Definition at line 192 of file nuller.c.

References `log_pedantic`, `mm_alloc()`, and `mm_copy()`.

Referenced by `cache_set_value()`, `config_value_set()`, `ip_subnet_st()`, `relay_set_value()`, and `servers_set_value()`.

5.73.1.9 `size_t ns_length_get (const chr_t *s)`

Return the length of a null-terminated string. LOW: The `ns_` functions are designed to work with NULL terminated strings. They shouldn't require a length. If the length needs to be provided, then the equivalent `mm_` function should be used.

Parameters:

s the input as a null-terminated string.

Returns:

the length in bytes of the string.

Definition at line 18 of file nuller.c.

Referenced by `api_endpoint_delete_user()`, `api_endpoint_register()`, `api_response()`, `build_version_major()`, `build_version_minor()`, `build_version_patch()`, `con_reverse_lookup()`, `con_write_ns()`, `config_fetch_host_number()`, `config_fetch_settings()`, `deprecated_ecies_key_private_hex()`, `deprecated_ecies_key_public_hex()`, `ecies_key_private_hex()`, `ecies_key_public_hex()`, `file_temp_handle()`, `host_platform()`, `host_version()`, `http_load_file()`, `ident_mt_mt()`, `imap_build_array()`, `imap_fetch_bodystructure()`, `imap_fetch_message()`, `imap_search_messages_date()`, `int16_conv_ns()`, `int32_conv_ns()`, `int64_conv_ns()`, `int8_conv_ns()`, `ip_addr_st()`, `ip_presentation()`, `ip_subnet_st()`, `lib_load_openssl()`, `line_pl_ns()`, `log_rotate()`, `log_start()`, `mail_mime_get_media_type()`, `mail_path_finder()`, `meta_folders_name()`, `mt_get_length()`, `ns_append()`, `ns_dupe()`, `ns_empty()`, `ns_empty_out()`, `ns_length_int()`, `portal_endpoint()`, `portal_endpoint_attachments_add()`, `portal_endpoint_error()`, `portal_endpoint_folders_add()`, `portal_endpoint_folders_rename()`, `portal_endpoint_response()`, `portal_parse_json_str_array()`, `rand_choices()`, `register_data_insert_user()`, `register_print_step1()`, `register_print_step2()`, `servers_network_start()`, `smtp_client_send_helo()`, `spool_path()`, `st_merge_opts()`, `stmt_rebuild()`, `stmt_start()`, `tank_start()`, `teacher_add_cookie()`, `uint16_conv_ns()`, `uint32_conv_ns()`, `uint64_conv_ns()`, `uint8_conv_ns()`, `virus_check()`, `xml_get_xpath_int16()`, `xml_get_xpath_int32()`, `xml_get_xpath_int64()`, `xml_get_xpath_int8()`, `xml_get_xpath_st()`, `xml_get_xpath_uint16()`, `xml_get_xpath_uint32()`, `xml_get_xpath_uint64()`, and `xml_get_xpath_uint8()`.

5.73.1.10 `int ns_length_int (chr_t *s)`

Return the length of a null-terminated string as an int, capped at INT_MAX.

Parameters:

s the null-terminated string to be evaluated.

Returns:

the length of the string.

Definition at line 51 of file nuller.c.

References `log_pedantic`, and `ns_length_get()`.

Referenced by `lib_load_bzip()`.

5.73.1.11 `bool_t ns_populated_variadic (ssize_t len, ...)`

A simple method for checking multiple NULL terminated strings to ensure all of the provided pointers lead to a buffer with at least one non-NULL character.

Parameters:

len the number of pointers being passed in.

va_list the list of NULL terminated string pointers to validate.

Returns:

true if none of the pointers are NULL, and all point to at least one non-NULL byte, otherwise false.

Definition at line 115 of file nuller.c.

5.73.1.12 `void ns_wipe (chr_t * s, size_t len)`

Zero out a null-terminated string.

Parameters:

s the string to be wiped.

len the number of bytes to be wiped at the start of the string.

Returns:

This function returns no parameters.

Definition at line 246 of file nuller.c.

References `log_pedantic`, and `mm_set()`.

Referenced by `deprecated_ecies_key_private_hex()`, `ecies_key_private_hex()`, and `ns_alloc()`.

5.74 src/core/strings/opts.c File Reference

```
#include "magma.h"
```

Functions

- `uint32_t st_opt_get (stringer_t *s)`
Returns the options variable from a string to the caller. Is intended for use by functions outside of string library.
- `int_t st_opt_set (stringer_t *s, uint32_t opt, bool_t enabled)`
Enable or disable a set of option(s) for a managed string, with validity testing.
- `bool_t st_opt_test (stringer_t *s, uint32_t opt)`
Check to see if the managed string has the specified option enabled.

5.74.1 Function Documentation

5.74.1.1 uint32_t st_opt_get (stringer_t * s)

Returns the options variable from a string to the caller. Is intended for use by functions outside of string library. [opts.c](#)

Parameters:

s the input string.

Returns:

the options in use by the provided string.

Definition at line 15 of file `opts.c`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `chunk_header_write()`, `ed25519_private_get()`, `ed25519_public_get()`, `ed25519_sign()`, `meta_data_fetch_shard()`, `org_key_get()`, `org_signet_get()`, `prime_field_write()`, `prime_header_write()`, `rand_choices()`, `secp256k1_compute_kek()`, `secp256k1_private_get()`, `secp256k1_public_get()`, `st_write_variadic()`, `stacie_create_nonce()`, `stacie_create_salt()`, `stacie_create_shard()`, `user_key_get()`, `user_request_get()`, and `user_signet_get()`.

5.74.1.2 int_t st_opt_set (stringer_t * s, uint32_t opt, bool_t enabled)

Enable or disable a set of option(s) for a managed string, with validity testing.

Parameters:

- s* the input managed string.
- opt* the bitmask of option(s) to be enabled or disabled for the managed string.
- enabled* if true, set the option(s); if false, disable them.

Returns:

-1 on error, 0 if successfully disabled, or 1 if successfully enabled.

Definition at line 33 of file `opts.c`.

References `log_pedantic`, `MEMORYBUF`, `st_info_opts()`, and `st_valid_opts()`.

Referenced by `contact_name()`.

5.74.1.3 bool_t st_opt_test (stringer_t * s, uint32_t *opt*)

Check to see if the managed string has the specified option enabled.

Parameters:

s the managed string to be checked. *opt* a bitmask of the managed string options to be tested.

Returns:

-1 on error, 0 if *opt* is not set, and 1 if *opt* is set.

Definition at line 64 of file `opts.c`.

References `log_pedantic`, `MEMORYBUF`, `st_info_opts()`, and `st_valid_opts()`.

Referenced by `contact_free()`, and `contact_name()`.

5.75 src/core/strings/print.c File Reference

```
#include "magma.h"
```

Functions

- `size_t st_vsprint (stringer_t *s, chr_t *format, va_list args)`
Print to a managed string and return the number of bytes written.
- `size_t st_sprint (stringer_t *s, chr_t *format,...)`
Print to a managed string and return the number of bytes written.
- `stringer_t * st_quick (stringer_t *s, chr_t *format,...)`
Print to a managed string and return a pointer to the result.
- `stringer_t * st_vaprint_opts (uint32_t opts, chr_t *format, va_list args)`
Return a managed string containing sprintf()-style formatted data.
- `stringer_t * st_aprint (chr_t *format,...)`
Return a managed string containing sprintf()-style formatted data.
- `stringer_t * st_aprint_opts (uint32_t opts, chr_t *format,...)`
Return a managed string containing sprintf()-style formatted data, with custom allocation options.
- `int_t st_write_variadic (stringer_t *output, ssize_t count,...)`
Write a variable number of user-supplied strings into a buffer.

5.75.1 Function Documentation

5.75.1.1 stringer_t* st_aprint (chr_t *format, ...)

Return a managed string containing sprintf()-style formatted data.

See also:

[st_vaprint_opts\(\)](#)

Parameters:

- format* a format string for the output string data.
- ...* a variable argument list of parameters to be formatted as output.

Returns:

NULL on failure or a managed string containing the final formatted data on success.

Definition at line 145 of file print.c.

References CONTIGUOUS, HEAP, MANAGED_T, and st_vaprint_opts().

Referenced by portal_endpoint_attachments_progress(), portal_endpoint_search(), portal_message_attachments(), register_captcha_random_font(), serial_get(), serial_increment(), serial_reset(), and spool_mktemp().

5.75.1.2 `stringer_t* st_aprint_opts (uint32_t opts, chr_t *format, ...)`

Return a managed string containing sprintf()-style formatted data, with custom allocation options.

Parameters:

opts the option value of the newly allocated managed string that will contain the result.

format a format string for the output string data.

... a variable argument list of parameters to be formatted as output.

Returns:

NULL on failure or a managed string of the specified allocation options containing the final formatted data on success.

Definition at line 164 of file print.c.

References `st_vaprint_opts()`.

Referenced by `http_response_cookie()`, `imap_fetch_message()`, and `imap_search()`.

5.75.1.3 `stringer_t* st_quick (stringer_t *s, chr_t *format, ...)`

Print to a managed string and return a pointer to the result.

See also:

`st_print()`

Parameters:

s a pointer to the managed string that will receive the output of the print operation.

format a format string specifying the arguments of the print operation.

... a variable argument list containing the parameters to the print format string.

Returns:

a pointer to the managed string that received the printed output.

Definition at line 82 of file print.c.

References `st_vsprintf()`.

Referenced by `api_endpoint_auth()`, `http_response_allow_cross()`, `http_response_cookie()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, `smtp_auth_plain()`, and `smtp_client_send_data()`.

5.75.1.4 `size_t st_sprint (stringer_t *s, chr_t *format, ...)`

Print to a managed string and return the number of bytes written.

Parameters:

s a pointer to the managed string that will receive the output of the print operation.

format a format string specifying the arguments of the print operation.

... a variable argument list containing the parameters to the print format string.

Returns:

-1 on failure, or the number of characters printed to the string, excluding the terminating null byte.

Definition at line 62 of file print.c.

References `st_vsprintf()`.

Referenced by `contact_abuse_checks()`, `contact_abuse_increment_history()`, `host_platform()`, `host_version()`, `imap_id()`, `imap_search()`, `ip_reversed()`, `ip_standard()`, `ip_subnet()`, `lock_get()`, `lock_release()`, `mail_build_signature()`, `register_data_insert_user()`, `smtp_bounce()`, `smtp_check_greylist()`, `smtp_reply()`, `tls_cipher()`, `tls_error()`, `user_lock()`, and `user_unlock()`.

5.75.1.5 `stringer_t* st_vaprint_opts (uint32_t opts, chr_t *format, va_list args)`

Return a managed string containing `sprintf()`-style formatted data.

Parameters:

opts an options value to be passed to the allocation of the resulting managed string.

format a format string for the output string data.

args a variable argument list of parameters to be formatted as output.

Returns:

NULL on failure or a managed string containing the final formatted data on success.

Definition at line 100 of file print.c.

References `length`, `log_pedantic`, `MEMORYBUF`, `st_alloc_opts()`, `st_data_get()`, `st_free()`, `st_info_opts()`, `st_length_set()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `st_aprint()`, and `st_aprint_opts()`.

5.75.1.6 `size_t st_vsprintf (stringer_t *s, chr_t *format, va_list args)`

Print to a managed string and return the number of bytes written.

See also:

`vsnprintf()`

Note:

If the destination string pointer is NULL the function will simply return how much room would have been necessary.

Parameters:

s a pointer to the managed string that will receive the output of the print operation.

format a format string specifying the arguments of the print operation.

args a `va_list` containing the parameters to the print format string.

Returns:

-1 on failure, or the number of characters printed to the string, excluding the terminating null byte.

Definition at line 19 of file print.c.

References `length`, `log_pedantic`, `MEMORYBUF`, `st_avail_get()`, `st_data_get()`, `st_info_opts()`, `st_length_set()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `st_quick()`, and `st_sprint()`.

5.75.1.7 `int_t st_write_variadic (stringer_t * output, ssize_t count, ...)`

Write a variable number of user-supplied strings into a buffer.

Parameters:

... a variable argument list containing the strings to be written one after the other.

Returns:

the number of bytes written into the buffer.

Definition at line 182 of file `print.c`.

References `data`, `mm_copy()`, `st_avail_get()`, `st_data_get()`, `st_empty_out()`, `st_length_set()`, `st_opt_get()`, and `st_valid_tracked()`.

5.76 src/core/strings/replace.c File Reference

```
#include "magma.h"
```

Functions

- [int_t st_replace](#) ([stringer_t](#) **target, [stringer_t](#) *pattern, [stringer_t](#) *replacement)
[replace.c](#)
- [stringer_t * st_swap](#) ([stringer_t](#) *target, [uchr_t](#) pattern, [uchr_t](#) replacement)
Replace all instances of one character in a managed string with another.

5.76.1 Function Documentation

5.76.1.1 [int_t st_replace](#) ([stringer_t](#) **target, [stringer_t](#) *pattern, [stringer_t](#) *replacement)

[replace.c](#)

Definition at line 19 of file [replace.c](#).

References [log_pedantic](#), [PLACER](#), [st_alloc\(\)](#), [st_char_get\(\)](#), [st_cmp_cs_starts\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), [st_length_set\(\)](#), and [st_valid_free\(\)](#).

Referenced by [contact_business\(\)](#), [imap_folder_create\(\)](#), [imap_folder_name_escaped\(\)](#), [imap_folder_rename\(\)](#), [pop_retr\(\)](#), [pop_top\(\)](#), [register_print_message\(\)](#), [register_print_step1\(\)](#), [register_print_step2\(\)](#), [smtp_client_send_data\(\)](#), [smtp_forward_message\(\)](#), [smtp_relay_message\(\)](#), and [teacher_process\(\)](#).

5.76.1.2 [stringer_t* st_swap](#) ([stringer_t](#) *target, [uchr_t](#) pattern, [uchr_t](#) replacement)

Replace all instances of one character in a managed string with another.

Parameters:

- target** the input string which will be transformed by the character replacement.
- pattern** the character to be searched and replaced in the target string.
- replacement** the character to be substituted for the pattern character in the target string.

Returns:

- a pointer to the target managed string.

Definition at line 109 of file [replace.c](#).

References [log_check](#), [log_pedantic](#), and [st_empty_out\(\)](#).

Referenced by [process_kill\(\)](#).

5.77 src/core/strings/shortcuts.c File Reference

```
#include "magma.h"
```

Functions

- [placer_t pl_null](#) (void)
Return a zero-length placer pointing to NULL data.
- [placer_t pl_init](#) (void *data, size_t len)
Return a placer wrapping a data buffer of given size.
- [placer_t pl_clone](#) (placer_t place)
- [placer_t pl_set](#) (placer_t place, placer_t set)
- void * [pl_data_get](#) (placer_t place)
Get a pointer to the data referenced by a placer.
- [chr_t * pl_char_get](#) (placer_t place)
Get a character pointer to the data referenced by a placer.
- [int_t pl_length_int](#) (placer_t place)
Get the length, in bytes, of a placer as an integer.
- size_t [pl_length_get](#) (placer_t place)
Get the length, in bytes, of a placer.
- [bool_t pl_empty](#) (placer_t place)
Determine whether or not the specified placer is empty.
- [bool_t pl_starts_with_char](#) (placer_t place, chr_t c)
Determine if a placer begins with a specified character.
- [bool_t pl_inc](#) (placer_t *place, bool_t more)
Advance the placer one character forward beyond an expected character.

5.77.1 Function Documentation

5.77.1.1 chr_t* pl_char_get (placer_t place)

Get a character pointer to the data referenced by a placer. [shortcuts.c](#)

Parameters:

place the input placer.

Returns:

NULL on failure or a a character pointer to the block of data associated with the specified placer on success.

Definition at line 54 of file shortcuts.c.

References [st_char_get\(\)](#).

Referenced by `cache_config()`, `con_write_pl()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_public()`, `dmtplib_process()`, `ecies_key_private()`, `ecies_key_public()`, `imap_fetch_body_header()`, `imap_narrow_messages()`, `imap_search_messages_range()`, `ip_subnet_st()`, `lib_load_openssl()`, `line_pl_pl()`, `molten_parse()`, `nvp_parse()`, `pl_get_embraced()`, `pl_shrink_before_characters()`, `pl_skip_characters()`, `pl_skip_to_characters()`, `pl_starts_with_char()`, `pl_trim()`, `pl_trim_end()`, `pl_trim_start()`, `pop_process()`, `portal_message_body()`, `portal_upload()`, `relay_config()`, `servers_config()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_client_send_data()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, `smtp_parse_rcpt_to()`, `smtp_process()`, and `str_tok_get_bl()`.

5.77.1.2 `placer_t pl_clone (placer_t place)`

Definition at line 30 of file `shortcuts.c`.

References `pl_init()`.

Referenced by `prime_pem_unwrap()`.

5.77.1.3 `void* pl_data_get (placer_t place)`

Get a pointer to the data referenced by a placer.

Parameters:

place the input placer.

Returns:

NULL on failure or a pointer to the block of data associated with the specified placer on success.

Definition at line 44 of file `shortcuts.c`.

References `st_data_get()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_public()`, `ecies_key_private()`, `ecies_key_public()`, `ephemeral_chunk_set()`, `imap_build_array()`, `imap_build_array_isliteral()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_parse_address_part()`, `nvp_parse()`, `org_signet_set()`, `prime_pem_unwrap()`, `smtp_check_filters()`, `tok_get_pl()`, `uint64_conv_pl()`, `user_request_set()`, and `user_signet_set()`.

5.77.1.4 `bool_t pl_empty (placer_t place)`

Determine whether or not the specified placer is empty.

Parameters:

place the input placer.

Returns:

true if the placer is empty or zero-length, or false otherwise.

Definition at line 84 of file `shortcuts.c`.

References `st_empty`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `cache_config()`, `client_read_line()`, `con_read_line()`, `dmtplib_process()`, `http_data_value_parse()`, `http_process()`, `imap_build_array()`, `imap_build_array_isliteral()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_narrow_messages()`, `imap_parse_address()`, `imap_parse_literal()`, `imap_process()`, `imap_search_messages_date()`, `imap_search_messages_header()`, `imap_search_messages_range()`, `mail_add_forward_headers()`, `mail_add_outbound_headers()`, `mail_mime_child()`, `mail_mime_count()`, `mail_mod_subject()`, `molten_parse()`, `multipart_get_boundary()`, `nvp_parse()`, `pl_get_embraced()`, `pl_inc()`, `pl_shrink_before_characters()`, `pl_skip_characters()`, `pl_skip_to_characters()`, `pl_starts_with_char()`, `pop_process()`, `portal_upload()`, `prime_pem_unwrap()`, `relay_config()`, `servers_config()`, `slots_key()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, `smtp_parse_rcpt_to()`, and `smtp_process()`.

5.77.1.5 `bool_t pl_inc (placer_t * place, bool_t more)`

Advance the placer one character forward beyond an expected character.

Parameters:

place the input placer.

more if true, the placer must contain more data, and vice versa.

Returns:

true if more was true and the placer contains more data, or if more was false and the placer ended; false otherwise.

Definition at line 114 of file shortcuts.c.

References `pl_empty()`.

5.77.1.6 `placer_t pl_init (void * data, size_t len)`

Return a placer wrapping a data buffer of given size.

Parameters:

data a pointer to the data to be wrapped.

len the length, in bytes, of the data.

Returns:

a placer pointing to the specified data.

Definition at line 25 of file shortcuts.c.

References `FOREIGNDATA`, `JOINTED`, `PLACER_T`, `placer_t`, and `STACK`.

Referenced by `aes_cipher_key()`, `aes_tag_shard()`, `aes_vector_shard()`, `api_endpoint_register()`, `bracket_extract_pl()`, `chunk_header_read()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `ecies_decrypt()`, `ecies_encrypt()`, `ephemeral_chunk_set()`, `get_header_value_noopt()`, `http_parse_context()`, `http_parse_origin()`, `http_parse_pairs()`, `imap_build_array()`, `imap_fetch_body()`, `imap_fetch_body_portion()`, `imap_fetch_bodystructure()`, `imap_fetch_envelope()`, `imap_parse_address_breaker()`, `imap_search_messages_date()`, `imap_search_messages_header()`, `line_pl_bl()`, `mail_get_chunk()`, `mail_header_pop()`, `mail_mime_child()`, `mail_mime_header()`, `mail_mime_part()`, `mail_mime_update()`, `mail_store_header()`, `naked_message_get()`, `naked_message_set()`, `pl_clone()`, `pl_trim()`, `pl_trim_end()`, `pl_trim_start()`, `prime_pem_unwrap()`, `prime_reader_open()`, `prime_reader_payload()`, `signature_full_verify()`, `signature_tree_verify()`, `slots_alloc()`, `slots_get()`, `smtp_check_filters()`, `smtp_client_send_helo()`, `str_tok_get_bl()`, `str_tok_get_count_bl()`, `tok_get_ns()`, and `tok_pop()`.

5.77.1.7 `size_t pl_length_get (placer_t place)`

Get the length, in bytes, of a placer.

Parameters:

place the input placer.

Returns:

the size, in bytes, of the specified placer.

Definition at line 74 of file shortcuts.c.

References `st_length_get()`.

Referenced by `cache_config()`, `chunk_buffer_read()`, `chunk_header_read()`, `client_read()`, `client_read_line()`, `con_read()`, `con_read_line()`, `con_write_pl()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_public()`, `dmtp_process()`, `ecies_key_private()`, `ecies_key_public()`, `imap_build_array()`, `imap_build_array_isliteral()`, `imap_fetch_body()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_parse_address_part()`, `imap_parse_literal()`, `ip_subnet_st()`, `line_pl_pl()`, `molten_parse()`, `nvp_parse()`, `org_signet_set()`, `pl_shrink_before_characters()`, `pl_trim()`, `pl_trim_end()`, `pl_trim_start()`, `pop_process()`, `portal_message_body()`, `portal_upload()`, `prime_pem_unwrap()`, `relay_config()`, `servers_config()`, `slots_buffer()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_check_filters()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, `smtp_parse_rcpt_to()`, `smtp_process()`, `str_tok_get_bl()`, `str_tok_get_count_bl()`, `tok_get_pl()`, `uint64_conv_pl()`, `user_request_set()`, and `user_signet_set()`.

5.77.1.8 `int_t pl_length_int (placer_t place)`

Get the length, in bytes, of a placer as an integer.

Parameters:

place the input placer.

Returns:

the size, in bytes, of the specified placer.

Definition at line 64 of file `shortcuts.c`.

References `st_length_int()`.

Referenced by `lib_load_openssl()`, `smtp_client_send_data()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, and `smtp_parse_rcpt_to()`.

5.77.1.9 `placer_t pl_null (void)`

Return a zero-length placer pointing to NULL data.

Returns:

a zero-length placer pointing to NULL data.

Definition at line 14 of file `shortcuts.c`.

References `FOREIGNDATA`, `JOINTED`, `PLACER_T`, `placer_t`, and `STACK`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `aes_cipher_key()`, `aes_tag_shard()`, `aes_vector_shard()`, `bracket_extract_pl()`, `build_version_major()`, `build_version_minor()`, `build_version_patch()`, `client_read()`, `client_read_line()`, `con_init_network_buffer()`, `con_read()`, `con_read_line()`, `get_header_opt()`, `imap_fetch_body()`, `imap_fetch_body_portion()`, `imap_fetch_envelope()`, `imap_narrow_messages()`, `imap_parse_address_breaker()`, `imap_parse_literal()`, `imap_search_messages_date()`, `imap_search_messages_header()`, `imap_search_messages_range()`, `imap_starttls()`, `line_pl_bl()`, `mail_get_chunk()`, `mail_header_pop()`, `mail_mime_child()`, `mail_modify_part()`, `naked_message_get()`, `naked_message_set()`, `nvp_parse()`, `part_decrypt()`, `pl_trim()`, `pl_trim_end()`, `pl_trim_start()`, `pop_starttls()`, `prime_pem_unwrap()`, `prime_reader_payload()`, `slots_buffer()`, `smtp_check_filters()`, `smtp_client_send_helo()`, `smtp_rcpt_to()`, `smtp_starttls()`, `str_tok_get_bl()`, `tok_get_ns()`, and `tok_pop()`.

5.77.1.10 `placer_t pl_set (placer_t place, placer_t set)`

Definition at line 34 of file `shortcuts.c`.

References `placer_t`.

Referenced by `mail_mime_split()`.

5.77.1.11 bool_t pl_starts_with_char (placer_t *place*, chr_t *c*)

Determine if a placer begins with a specified character.

Parameters:

place the input placer.

c the character to be compared with the first byte of the placer's data.

Returns:

true if the placer begins with the given character or false otherwise.

Definition at line 95 of file shortcuts.c.

References `pl_char_get()`, and `pl_empty()`.

Referenced by `nvp_parse()`, `prime_pem_unwrap()`, and `smtp_client_send_data()`.

5.78 src/core/strings/strings.h File Reference

Data Structures

- struct [multi_t](#)

Defines

- #define [CONSTANT](#)(string) ([stringer_t](#) *)(([constant_t](#) *)"\x41\x01\x00\x00" string)
- #define [NULLER](#)(d) ([stringer_t](#) *)&([nuller_t](#)) { .opts = (NULLER_T | JOINTED | STACK | FOREIGNDATA), .data = d }
- #define [BLOCK](#)(d, l) ([stringer_t](#) *)&([block_t](#)) { .opts = (BLOCK_T | JOINTED | STACK | FOREIGNDATA), .data = d, .length = l }
- #define [PLACER](#)(d, l) ([stringer_t](#) *)&([placer_t](#)) { .opts = (PLACER_T | JOINTED | STACK | FOREIGNDATA), .data = d, .length = l }
- #define [MANAGED](#)(d, l, a) ([stringer_t](#) *)&([managed_t](#)) { .opts = (MANAGED_T | JOINTED | STACK | FOREIGNDATA), .data = d, .length = l, .avail = a }
- #define [BLOCKBUF](#)(l) ([stringer_t](#) *)&([block_t](#)) { .opts = (BLOCK_T | CONTIGUOUS | STACK), .data = &(([chr_t](#) []){ [0 ... l] = 0 }), .length = l }
- #define [MANAGEDBUF](#)(l) ([stringer_t](#) *)&([managed_t](#)) { .opts = (MANAGED_T | CONTIGUOUS | STACK), .data = &(([chr_t](#) []){ [0 ... l] = 0 }), .length = 0, .avail = l }
- #define [st_append](#)(s, append) st_append_opts(1024, s, append)
- #define [st_merge](#)(...) st_merge_opts(MANAGED_T | CONTIGUOUS | HEAP, __VA_ARGS__)
- #define [st_vaprint](#)(format, args) st_vaprint_opts(MANAGED_T | CONTIGUOUS | HEAP, format, args)
- #define [st_empty](#)(...) st_empty_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)
- #define [st_populated](#)(...) st_populated_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)
- #define [ns_populated](#)(...) ns_populated_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)
- #define [st_cleanup](#)(...) st_cleanup_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)
- #define [ns_cleanup](#)(...) ns_cleanup_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)
- #define [st_write](#)(output,...) st_write_variadic(output, va_narg(__VA_ARGS__), ##__VA_ARGS__)
- #define [va_narg](#)(...) (__VA_NARG__(_0, ##__VA_ARGS__, __VA_NARG_SEQ_N()) - 1)
- #define [__VA_NARG__](#)(...) __VA_NARG_N(__VA_ARGS__)
- #define [__VA_NARG_N](#)(_1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, _16, _17, _18, _19, _20, _21, _22, _23, _24, _25, _26, _27, _28, _29, _30, _31, _32, _33, _34, _35, _36, _37, _38, _39, _40, _41, _42, _43, _44, _45, _46, _47, _48, _49, _50, _51, _52, _53, _54, _55, _56, _57, _58, _59, _60, _61, _62, _63, N,...) N
- #define [__VA_NARG_SEQ_N](#)()

Typedefs

- typedef void [stringer_t](#)

Enumerations

- enum {
[CONSTANT_T](#) = 1, [PLACER_T](#) = 2, [NULLER_T](#) = 4, [BLOCK_T](#) = 8,
[MANAGED_T](#) = 16, [MAPPED_T](#) = 32, [CONTIGUOUS](#) = 64, [JOINTED](#) = 128,
[STACK](#) = 256, [HEAP](#) = 512, [SECURE](#) = 1024, [FOREIGNDATA](#) = 4096 }

Functions

- `struct __attribute__((packed))`
- `chr_t * ns_alloc (size_t len)`
nuller.c
- `chr_t * ns_append (chr_t *s, chr_t *append)`
Append one string to another and return the result.
- `chr_t * ns_dupe (chr_t *s)`
Duplicate a null-terminated string.
- `bool_t ns_empty (chr_t *s)`
Return whether a null-terminated string is empty or not.
- `bool_t ns_empty_out (chr_t *s, chr_t **ptr, size_t *len)`
Return whether a null-terminated string is empty or not, while storing its address and length.
- `void ns_free (chr_t *s)`
Free a null-terminated string.
- `bool_t ns_populated_variadic (ssize_t len,...)`
A simple method for checking multiple NULL terminated strings to ensure all of the provided pointers lead to a buffer with at least one non-NULL character.
- `void ns_cleanup_variadic (ssize_t len,...)`
A checked cleanup function which can be used free a variable number of null-terminated strings.
- `chr_t * ns_import (void *block, size_t len)`
Get a block of memory as a null-terminated string.
- `size_t ns_length_get (const chr_t *s)`
Return the length of a null-terminated string.
- `int ns_length_int (chr_t *s)`
Return the length of a null-terminated string as an int, capped at INT_MAX.
- `void ns_wipe (chr_t *s, size_t len)`
Zero out a null-terminated string.
- `const chr_t * st_info_type (uint32_t opts)`
Get a readable description of a managed string's data type.
- `const chr_t * st_info_layout (uint32_t opts)`
Get a readable description of a managed string's data layout.
- `const chr_t * st_info_allocator (uint32_t opts)`
Get a readable description of a managed string's allocator type.
- `chr_t * st_info_opts (uint32_t opts, chr_t *s, size_t len)`
Get a readable description of all of a managed string's option flags.
- `bool_t st_valid_free (uint32_t opts)`

Determine whether a managed string is allowed to be freed.

- `bool_t st_valid_opts` (uint32_t opts)

Check to see that a managed string has a valid combination of allocation options.

- `bool_t st_valid_avail` (uint32_t opts)

Determine whether a managed string tracks the total allocated buffer size.

- `bool_t st_valid_append` (uint32_t opts)

Determine whether the managed string options allow for data appending.

- `bool_t st_valid_placer` (uint32_t opts)

A sanity check to determine whether the managed string is a valid placer.

- `bool_t st_valid_jointed` (uint32_t opts)

Determine whether the managed string options are a validly jointed.

- `bool_t st_valid_tracked` (uint32_t opts)

Determine whether a managed string provides for data length tracking.

- `bool_t st_valid_destination` (uint32_t opts)

Determine whether the managed string options allow for a data store operation.

- `int_t st_length_int` (stringer_t *s)

Length.

- `size_t st_avail_get` (stringer_t *s)

Return the total data buffer size allocated for a managed string.

- `size_t st_length_get` (stringer_t *s)

Return the length of the data in a managed string.

- `size_t st_avail_set` (stringer_t *s, size_t avail)

Set the total data buffer size allocated for a managed string.

- `size_t st_length_set` (stringer_t *s, size_t len)

Set the data length for a managed string that supports data length tracking.

- `chr_t * st_char_get` (stringer_t *s)

data.c

- `void * st_data_get` (stringer_t *s)

Retrieve the data associated with a managed string.

- `void st_data_set` (stringer_t *s, void *data)

Set the underlying data of a jointed managed string.

- `bool_t st_empty_out` (stringer_t *s, uchr_t **ptr, size_t *len)

Determine whether the specified managed string is empty or not, and store its underlying data and data length.

- `bool_t st_empty_variadic` (ssize_t len,...)

A simple method for checking multiple managed strings to see if any are empty.

- [bool_t st_populated_variadic](#) (ssize_t len,...)
A simple method for checking multiple managed strings to ensure all of the provided strings have contain at least one character of data.
- [bool_t st_used_variadic](#) (ssize_t len,...)
- [uchar_t * st_uchar_get](#) (stringer_t *s)
Retrieve an unsigned character pointer to a managed string's data buffer.
- void [st_wipe](#) (stringer_t *s)
Wipe all of a managed string's allocated memory and if applicable, reset its length.
- [stringer_t * st_set](#) (stringer_t *s, uint8_t set, size_t len)
Sets a block of memory to a specified value.
- void [st_free](#) (stringer_t *s)
Free a managed string.
- void [st_cleanup_variadic](#) (ssize_t len,...)
A checked cleanup function which can be used free a variable number managed strings.
- [stringer_t * st_alloc](#) (size_t len)
Allocate a managed string with a specified options mask.
- [stringer_t * st_dupe](#) (stringer_t *s)
Duplicate a managed string.
- [stringer_t * st_import](#) (const void *s, size_t len)
Create a new (contiguous managed) managed string on the heap to hold a copy of the specified data buffer.
- [stringer_t * st_copy_in](#) (stringer_t *s, void *buf, size_t len)
Copy data into a managed string.
- [stringer_t * st_realloc](#) (stringer_t *s, size_t len)
Reallocate a managed string to a specified size.
- [stringer_t * st_output](#) (stringer_t *output, size_t len)
Return a suitable managed string to store data of a specified length.
- [stringer_t * st_nullify](#) (chr_t *input, size_t len)
Create a null-terminated string out of a block of memory and return it as a newly allocated nuller.
- [stringer_t * st_import_opts](#) (uint32_t opts, const void *s, size_t len)
Create a new (contiguous managed) managed string on the heap to hold a copy of the specified data buffer.
- [stringer_t * st_alloc_opts](#) (uint32_t opts, size_t len)
Allocate a managed string with a specified options mask.
- [stringer_t * st_dupe_opts](#) (uint32_t opts, stringer_t *s)
Create a duplicate copy of a managed string with a specified set of allocation options.
- [stringer_t * st_merge_opts](#) (uint32_t opts, chr_t *format,...)
Concatenate a variable number of user-supplied multi-type strings.
- [stringer_t * st_append_opts](#) (size_t align, stringer_t *s, stringer_t *append)

Append one managed string to another, with aligned memory boundaries.

- `int_t st_append_out` (`size_t align`, `stringer_t **s`, `stringer_t *append`)
- `chr_t * pl_char_get` (`placer_t place`)

shortcuts.c

- `void * pl_data_get` (`placer_t place`)

Get a pointer to the data referenced by a placer.

- `bool_t pl_empty` (`placer_t place`)

Determine whether or not the specified placer is empty.

- `placer_t pl_init` (`void *data`, `size_t len`)

Return a placer wrapping a data buffer of given size.

- `placer_t pl_clone` (`placer_t place`)
- `size_t pl_length_get` (`placer_t place`)

Get the length, in bytes, of a placer.

- `int_t pl_length_int` (`placer_t place`)

Get the length, in bytes, of a placer as an integer.

- `placer_t pl_null` (`void`)

Return a zero-length placer pointing to NULL data.

- `placer_t pl_set` (`placer_t place`, `placer_t set`)
- `bool_t pl_starts_with_char` (`placer_t place`, `chr_t c`)

Determine if a placer begins with a specified character.

- `bool_t pl_inc` (`placer_t *place`, `bool_t more`)

Advance the placer one character forward beyond an expected character.

- `uint32_t st_opt_get` (`stringer_t *s`)

opts.c

- `bool_t st_opt_test` (`stringer_t *s`, `uint32_t opt`)

Check to see if the managed string has the specified option enabled.

- `int_t st_opt_set` (`stringer_t *s`, `uint32_t opt`, `bool_t enabled`)

Enable or disable a set of option(s) for a managed string, with validity testing.

- `stringer_t * st_aprint` (`chr_t *format`,...) `__attribute__((format(printf`

print.c

- `stringer_t stringer_t * st_aprint_opts` (`uint32_t opts`, `chr_t *format`,...) `__attribute__((format(printf`
- `stringer_t stringer_t stringer_t * st_quick` (`stringer_t *s`, `chr_t *format`,...) `__attribute__((format(printf`
- `stringer_t stringer_t stringer_t size_t st_sprint` (`stringer_t *s`, `chr_t *format`,...) `__attribute__((format(printf`
- `stringer_t stringer_t stringer_t size_t st_vaprint_opts` (`uint32_t opts`, `chr_t *format`, `va_list args`)

Return a managed string containing sprintf()-style formatted data.

- `size_t st_vsprint` (`stringer_t *s`, `chr_t *format`, `va_list args`)

Print to a managed string and return the number of bytes written.

- `int_t st_write_variadic` (`stringer_t` *output, `ssize_t` count,...)
Write a variable number of user-supplied strings into a buffer.
- `int_t st_replace` (`stringer_t` **target, `stringer_t` *pattern, `stringer_t` *replacement)
replace.c
- `stringer_t * st_swap` (`stringer_t` *target, `uchr_t` pattern, `uchr_t` replacement)
Replace all instances of one character in a managed string with another.
- `int32_t cmp_mt_mt` (`multi_t` one, `multi_t` two)
multi.c
- `bool_t ident_mt_mt` (`multi_t` one, `multi_t` two)
Check to see if the values of two multi-type objects are identical.
- `multi_t mt_dupe` (`multi_t` multi)
Duplicate a multi-type object.
- `void mt_free` (`multi_t` multi)
Free the data underlying a multi-type data object (for managed and null-terminated strings).
- `char * mt_get_char` (const `multi_t` *multi)
Get a character pointer to the value of a multi-type object.
- `size_t mt_get_length` (`multi_t` multi)
Get the length of the data associated with a multi-type object.
- `multi_t mt_get_null` (void)
Return an empty multi-type object.
- `uint64_t mt_get_number` (`multi_t` multi)
Get the value of a numerical multi-type object.
- `M_TYPE mt_get_type` (`multi_t` multi)
Get the data type associated with a multi-type object.
- `bool_t mt_is_empty` (`multi_t` multi)
Determine whether a multi-type object is empty.
- `bool_t mt_is_number` (`multi_t` multi)
Determine whether a multi-type object is a number.
- `multi_t mt_set_type` (`multi_t` multi, `M_TYPE` target)
Set the data type of a multi-type data object.

Variables

- `constant_t`
- `nuller_t`
- `block_t`
- `placer_t`
- `managed_t`
- `mapped_t`

5.78.1 Define Documentation

5.78.1.1 #define __VA_NARG__(...) __VA_NARG_N(__VA_ARGS__)

Definition at line 262 of file strings.h.

5.78.1.2 #define __VA_NARG_N(_1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, _16, _17, _18, _19, _20, _21, _22, _23, _24, _25, _26, _27, _28, _29, _30, _31, _32, _33, _34, _35, _36, _37, _38, _39, _40, _41, _42, _43, _44, _45, _46, _47, _48, _49, _50, _51, _52, _53, _54, _55, _56, _57, _58, _59, _60, _61, _62, _63, N, ...) N

Definition at line 266 of file strings.h.

5.78.1.3 #define __VA_NARG_SEQ_N()

Value:

```
63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, \
 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 1
 5, 14, 13, 12, 11, 10, \
 9, 8, 7, 6, 5, 4, 3, 2, 1, 0
```

Definition at line 267 of file strings.h.

5.78.1.4 #define BLOCK(d, l) (stringer_t *)&((block_t){ .opts = (BLOCK_T | JOINTED | STACK | FOREIGNDATA), .data = d, .length = l })

Definition at line 188 of file strings.h.

5.78.1.5 #define BLOCKBUF(l) (stringer_t *)&((block_t){ .opts = (BLOCK_T | CONTIGUOUS | STACK), .data = &((chr_t []){ [0 ... l] = 0 }), .length = l })

Definition at line 197 of file strings.h.

5.78.1.6 #define CONSTANT(string) (stringer_t *)((constant_t *)"\x41\x01\x00\x00" string)

Definition at line 182 of file strings.h.

Referenced by cache_config(), cache_set_value(), config_load_cmdline_settings(), config_load_database_settings(), config_load_file_settings(), config_output_value_generic(), config_value_set(), file_temp_handle(), lib_load(), mail_add_forward_headers(), mail_add_inbound_headers(), mail_mod_subject(), meta_folders_by_name(), protocol_type(), relay_config(), relay_set_value(), sanity_check(), serial_reset(), servers_config(), servers_output_settings(), servers_set_value(), servers_validate(), smtp_parse_mail_from_path(), stats_sum_errors(), and virus_check().

5.78.1.7 #define MANAGED(d, l, a) (stringer_t *)&((managed_t){ .opts = (MANAGED_T | JOINTED | STACK | FOREIGNDATA), .data = d, .length = l, .avail = a })

Definition at line 194 of file strings.h.

Referenced by encrypted_chunk_set().

5.78.1.8 #define MANAGEDBUF(l) (stringer_t *)&((managed_t){ .opts = (MANAGED_T | CONTIGUOUS | STACK), .data = &((chr_t []){ [0 ... l] = 0 }), .length = 0, .avail = l })

Definition at line 200 of file strings.h.

Referenced by `_serialize_ec_pubkey()`, `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `api_endpoint_auth()`, `auth_legacy()`, `client_read()`, `client_read_line()`, `client_write()`, `contact_abuse_checks()`, `contact_abuse_increment_history()`, `contact_business()`, `deprecated_scramble_decrypt()`, `deprecated_scramble_encrypt()`, `encrypted_chunk_get()`, `encrypted_chunk_set()`, `ephemeral_chunk_get()`, `http_response_allow_cross()`, `http_response_cookie()`, `http_response_header()`, `http_response_options()`, `imap_id()`, `imap_login()`, `imap_search()`, `lib_load()`, `lock_get()`, `lock_release()`, `mail_add_inbound_headers()`, `mail_add_outbound_headers()`, `meta_data_insert_shard()`, `meta_update_realms()`, `naked_message_set()`, `org_key_get()`, `org_signet_fingerprint()`, `org_signet_generate()`, `org_signet_get()`, `org_signet_verify()`, `pop_pass()`, `portal_endpoint_auth()`, `portal_meta()`, `prime_pem_unwrap()`, `prime_pem_wrap()`, `process_find_pid()`, `process_kill()`, `protocol_secure()`, `register_abuse_check_blocklist()`, `register_abuse_checks()`, `register_abuse_increment_history()`, `register_data_insert_user()`, `register_session_cache()`, `register_session_get()`, `scramble_decrypt()`, `scramble_encrypt()`, `signature_full_get()`, `signature_full_verify()`, `signature_tree_get()`, `signature_tree_verify()`, `slots_key()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_bounce()`, `smtp_check_greylist()`, `smtp_check_rbl()`, `smtp_client_send_data()`, `smtp_rcpt_to()`, `smtp_reply()`, `stacie_decrypt()`, `stacie_encrypt()`, `tcp_status()`, `user_key_get()`, `user_lock()`, `user_request_generate()`, `user_request_get()`, `user_request_rotation()`, `user_request_sign()`, `user_request_verify_chain_of_custody()`, `user_request_verify_self()`, `user_signet_fingerprint()`, `user_signet_get()`, `user_signet_verify_chain_of_custody()`, `user_signet_verify_org()`, `user_signet_verify_self()`, and `user_unlock()`.

5.78.1.9 `#define ns_cleanup(...) ns_cleanup_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)`

Definition at line 252 of file `strings.h`.

5.78.1.10 `#define ns_populated(...) ns_populated_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)`

Definition at line 248 of file `strings.h`.

Referenced by `dkim_signature_create()`.

5.78.1.11 `#define NULLER(d) (stringer_t *)&((nuller_t){ .opts = (NULLER_T | JOINTED | STACK | FOREIGNDATA), .data = d })`

Definition at line 185 of file `strings.h`.

Referenced by `api_endpoint_auth()`, `api_endpoint_register()`, `api_response()`, `args_parse()`, `cache_config()`, `cmp_mt_mt()`, `color_supported()`, `config_free()`, `config_key_lookup()`, `config_output_value()`, `config_output_value_generic()`, `config_value_set()`, `contact_business()`, `contact_business_add_error()`, `contact_print_form()`, `contact_print_message()`, `contact_process()`, `folder_count()`, `http_content_load_directory()`, `http_content_load_fonts()`, `http_content_start()`, `http_data_get()`, `http_get_template()`, `http_load_file()`, `http_parse_context()`, `http_response()`, `imap_list()`, `imap_lsub()`, `imap_status()`, `ip_subnet_st()`, `lib_load()`, `log_start()`, `mail_mime_get_smtp_envelope()`, `multipart_get_boundary()`, `portal_endpoint_auth()`, `portal_endpoint_config_edit()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_error()`, `portal_endpoint_folders_add()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_remove()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_endpoint_response()`, `portal_parse_flags()`, `portal_parse_sections()`, `portal_upload()`, `prime_pem_wrap()`, `register_abuse_checks()`, `register_abuse_increment_history()`, `register_captcha_random_font()`, `register_print_message()`, `register_session_cache()`, `register_session_get()`, `relay_config()`, `sanity_check()`, `servers_config()`, `servers_output_settings()`, `servers_set_value()`, `servers_validate()`, `spool_check_file()`, `st_info_opts()`, `STACK_OF()`, `stats_get_name_pos()`, `stats_sum_errors()`, `tls_cipher()`, and `tls_version()`.

5.78.1.12 `#define PLACER(d, l) (stringer_t *)&((placer_t){ .opts = (PLACER_T | JOINTED | STACK | FOREIGNDATA), .data = d, .length = l })`

Definition at line 191 of file `strings.h`.

Referenced by `_deserialize_ec_privkey()`, `_load_ec_privkey()`, `_serialize_ec_privkey()`, `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `api_endpoint_register()`, `api_response()`, `args_parse()`, `auth_data_fetch()`, `auth_sanitise_address()`, `auth_sanitise_username()`, `build_version_major()`, `build_version_minor()`, `build_version_patch()`, `cache_config()`, `config_free()`, `config_load_database_settings()`, `config_output_value()`, `config_output_value_generic()`, `config_validate_settings()`, `config_value_set()`, `contact_business()`, `contact_business_add_error()`, `contact_details_fetch()`, `contact_edit()`, `contact_print_form()`, `contact_print_message()`, `contact_process()`, `contacts_fetch()`, `deprecated_scramble_decrypt()`, `deprecated_symmetric_vector()`, `dmtmp_compare()`, `encrypted_chunk_get()`,

encrypted_chunk_set(), ephemeral_chunk_set(), folder_count(), http_content_load_directory(), http_content_load_fonts(), http_content_start(), http_load_file(), http_parse_header(), http_parse_method(), http_parse_origin(), http_parse_pairs(), http_print_400(), http_print_403(), http_print_404(), http_print_405(), http_print_500(), http_print_500_log(), http_print_501(), http_response(), http_response_allow_cross(), http_response_connection(), http_response_cookie(), ident_mt_mt(), imap_command_log_safe(), imap_command_parser(), imap_compare(), imap_fetch(), imap_fetch_body(), imap_fetch_body_mime(), imap_fetch_bodystructure(), imap_fetch_envelope(), imap_fetch_message(), imap_flag_action(), imap_folder_compare(), imap_folder_create(), imap_folder_name_escaped(), imap_folder_remove(), imap_folder_rename(), imap_get_flag(), imap_list(), imap_lsub(), imap_parse_dataitems(), imap_search(), imap_search_messages_date(), imap_search_messages_date_compare(), imap_search_messages_inner(), imap_status(), json_api_dispatch(), lock_get(), magma_folder_fetch(), magma_folder_find_full_name(), magma_folder_find_name(), magma_folder_name(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_add_required_headers(), mail_count_received(), mail_discover_encoding(), mail_discover_insertion_point(), mail_discover_type(), mail_get_boundary(), mail_get_chunk(), mail_header_fetch_all(), mail_header_fetch_cleaned(), mail_headers(), mail_insert_chunk_base64(), mail_insert_chunk_text(), mail_message(), mail_mime_boundary(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_encoding(), mail_mime_type(), mail_mime_type_group(), mail_mime_type_parameters(), mail_mime_type_sub(), mail_mod_subject(), mail_modify_part(), mail_signature_add(), mail_store_message(), main(), meta_data_fetch_alerts(), meta_data_fetch_folder_messages(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_shard(), meta_data_fetch_user(), meta_update_realms(), molten_compare(), naked_message_get(), naked_message_set(), net_trigger(), part_decrypt(), part_encrypt(), pop_compare(), pop_retr(), pop_top(), portal_endpoint(), portal_endpoint_attachments_progress(), portal_endpoint_compare(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_cookies(), portal_endpoint_error(), portal_endpoint_folders_list(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_tag(), portal_endpoint_response(), portal_endpoint_search(), portal_message_attachments(), portal_message_body(), portal_message_header(), portal_parse_action(), portal_parse_context(), portal_parse_flags(), portal_parse_sections(), portal_process(), portal_upload(), prime_pem_wrap(), prime_unpack(), qp_encode(), register_business_step2(), register_captcha_random_font(), register_data_insert_user(), register_print_message(), register_print_step1(), register_print_step2(), relay_config(), scramble_decrypt(), servers_config(), sess_token(), signature_full_get(), signature_tree_get(), smtp_accept_message(), smtp_auth_login(), smtp_auth_plain(), smtp_check_filters(), smtp_client_close(), smtp_client_send_data(), smtp_compare(), smtp_data_outbound(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_forward_message(), smtp_get_action(), smtp_parse_mail_from_path(), smtp_relay_message(), smtp_update_receive_stats(), spool_check_file(), st_replace(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_decrypt(), stacie_encrypt(), stacie_realm_cipher(), stacie_realm_key(), stacie_realm_tag(), stacie_realm_vector(), stamp_counter_check(), stamp_counter_increment(), symmetric_vector(), teacher_data_get(), teacher_data_save(), teacher_process(), tls_cipher(), tls_version(), tran_commit(), tran_rollback(), tran_start(), url_encode(), user_config_fetch(), virus_check(), warehouse_fetch_domains(), xml_get_xpath_int16(), xml_get_xpath_int32(), xml_get_xpath_int64(), xml_get_xpath_int8(), xml_get_xpath_uint16(), xml_get_xpath_uint32(), xml_get_xpath_uint64(), and xml_get_xpath_uint8().

5.78.1.13 #define st_append(s, append) st_append_opts(1024, s, append)

Definition at line 239 of file strings.h.

Referenced by args_parse(), file_temp_handle(), http_response_allow_cross(), http_response_cookie(), magma_folder_name(), naked_message_set(), part_buffer(), part_decrypt(), qp_encode(), signature_tree_get(), signature_tree_verify(), st_info_opts(), and url_encode().

5.78.1.14 #define st_cleanup(...) st_cleanup_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)

Definition at line 251 of file strings.h.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), api_endpoint_register(), auth_free(), auth_legacy_free(), auth_login(), auth_stacie_free(), client_close(), con_destroy(), dkim_signature_create(), ed25519_sign(), http_body(), http_data_free(), http_free_content(), http_print_301(), http_response_header(), http_response_options(), http_session_reset(), imap_command_parser(), imap_fetch_body(), imap_fetch_body_header(), imap_fetch_body_mime(), imap_fetch_body_tag(), imap_fetch_bodystructure(), imap_fetch_envelope(), imap_fetch_response_free(), imap_folder_create(), imap_folder_rename(), imap_search(), imap_session_destroy(), keks_free(), magma_folder_find_full_name(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_add_required_headers(), mail_build_signature(), mail_cache_destroy(), mail_cache_reset(), mail_cache_set(), mail_destroy(), mail_destroy_header(), mail_destroy_message(), mail_get_boundary(), mail_insert_chunk_base64(), mail_mime_boundary(), mail_mime_free(), mail_mime_get_smtp_envelope(), mail_store_message(), message_free(), meta_crypto_keys_create(), meta_data_fetch_user(), meta_data_insert_keys(), meta_free(), meta_get(), meta_update_keys(), naked_message_set(), org_key_get(), org_signet_generate(), org_signet_get(), part_buffer(), part_decrypt(), pop_session_destroy(), pop_user(), portal_endpoint_contacts_copy(), portal_endpoint_messages_list(), portal_folder_contacts_add(), portal_folder_mail_add(), portal_folder_mail_remove(), portal_message_attachments(), portal_message_body(), portal_message_header(), portal_smtp_merge_headers(), prime_key_decrypt(), prime_set(), rand_choices(), register_business_step2(), register_data_insert_user(), register_session_cache(), register_session_free(), secp256k1_compute_kek(), secp256k1_private_

get(), secp256k1_public_get(), sess_destroy(), sess_get(), sess_release_attachment(), sess_token(), signature_full_get(), signature_tree_get(), smtp_auth_login(), smtp_auth_plain(), smtp_bounce(), smtp_check_filters(), smtp_check_greylist(), smtp_client_send_data(), smtp_data_outbound(), smtp_ehlo(), smtp_fetch_inbound(), smtp_free_inbound(), smtp_free_outbound(), smtp_free_recipients(), smtp_helo(), smtp_list_free_filter(), smtp_reply(), smtp_session_destroy(), smtp_session_reset(), st_info_opts(), st_merge_opts(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_realm_key(), teacher_data_delete(), teacher_data_free(), teacher_process(), user_key_get(), user_request_get(), and user_signet_get().

5.78.1.15 #define st_empty(...) st_empty_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)

Definition at line 244 of file strings.h.

Referenced by aes_cipher_key(), aes_tag_shard(), aes_vector_shard(), auth_challenge(), auth_data_fetch(), auth_data_update_legacy(), auth_legacy(), auth_login(), auth_response(), auth_stacie(), cache_add(), cache_append(), cache_decrement(), cache_delete(), cache_free(), cache_get(), cache_get_u64(), cache_increment(), cache_output_settings(), cache_set(), cache_set_u64(), cache_set_value(), cache_silent_add(), cache_validate(), cipher_name(), client_read_line(), compress_import(), con_read_line(), config_load_cmdline_settings(), config_load_database_settings(), config_load_file_settings(), config_output_value(), config_output_value_generic(), config_value_set(), contact_create(), contact_edit(), contact_find_detail(), contact_find_name(), contact_folder_create(), contact_folder_rename(), contact_update(), deprecated_hmac_digest(), deprecated_scramble_import(), digest_name(), dkim_signature_create(), double_conv(), ed25519_private_set(), ed25519_public_set(), ed25519_sign(), ed25519_verify(), float_conv(), folder_count(), hash_digest(), hex_decode_opts(), hex_encode_opts(), hmac_digest(), http_body(), lock_get(), lock_release(), magma_folder_find_full_name(), magma_folder_find_name(), magma_folder_name(), mail_add_outbound_headers(), mail_add_required_headers(), mail_header_fetch_all(), mail_header_fetch_cleaned(), mail_header_pop(), mail_insert_chunk_text(), mail_mime_child(), mail_mime_count(), mail_mime_part(), mail_mime_split(), message_folder_create(), meta_data_fetch_keys(), meta_data_fetch_shard(), meta_data_fetch_user(), meta_data_insert_shard(), meta_get(), meta_update_realms(), mt_is_empty(), pl_empty(), pop_pass(), prime_pem_wrap(), prime_reader_open(), relay_free(), relay_output_settings(), relay_set_value(), relay_validate(), scramble_import(), secp256k1_private_set(), secp256k1_public_set(), servers_free(), servers_network_start(), servers_output_settings(), servers_set_value(), servers_validate(), smtp_auth_login(), smtp_auth_plain(), smtp_client_send_data(), smtp_client_send_helo(), smtp_fetch_inbound(), spf_check(), st_bitwise(), st_info_opts(), st_merge_opts(), st_not(), stacie_realm_cipher(), stacie_realm_tag(), stacie_realm_vector(), uint16_get_no(), uint24_get_no(), uint32_get_no(), user_config_edit(), and user_request_sign().

5.78.1.16 #define st_merge(...) st_merge_opts(MANAGED_T | CONTIGUOUS | HEAP, __VA_ARGS__)

Definition at line 240 of file strings.h.

Referenced by auth_legacy(), contact_business(), dkim_signature_create(), ephemeral_chunk_get(), folder_count(), http_content_load_directory(), http_content_load_fonts(), http_print_301(), imap_build_array_isliteral(), imap_fetch_body(), imap_fetch_body_header(), imap_fetch_body_mime(), imap_fetch_body_tag(), imap_fetch_bodystructure(), imap_fetch_message(), imap_folder_create(), imap_folder_name_escaped(), imap_folder_rename(), imap_narrow_folders(), imap_parse_address(), imap_range_build(), imap_search_messages_date(), mail_add_required_headers(), mail_build_signature(), mail_get_boundary(), mail_header_fetch_all(), mail_insert_chunk_base64(), mail_insert_chunk_text(), mail_mime_boundary(), mail_mime_encode_part(), mail_mime_get_smtp_envelope(), mail_modify_subject(), meta_folders_name(), naked_message_get(), naked_message_set(), portal_endpoint_contacts_copy(), portal_folder_mail_add(), portal_message_attachments(), portal_smtp_create_data(), portal_smtp_merge_headers(), register_business_step2(), register_data_insert_user(), sess_token(), smtp_bounce(), and smtp_reply().

5.78.1.17 #define st_populated(...) st_populated_variadic(va_narg(__VA_ARGS__), ##__VA_ARGS__)

Definition at line 247 of file strings.h.

Referenced by api_endpoint_auth(), dkim_signature_create(), imap_login(), imap_search(), mail_modify_part(), meta_data_insert_keys(), meta_get(), meta_update_keys(), meta_update_user(), pop_pass(), portal_endpoint_auth(), smtp_auth_login(), smtp_auth_plain(), and smtp_fetch_authorization().

5.78.1.18 #define st_vaprint(format, args) st_vaprint_opts(MANAGED_T | CONTIGUOUS | HEAP, format, args)

Definition at line 241 of file strings.h.

Referenced by client_print().

5.78.1.19 #define st_write(output, ...) st_write_variadic(output, va_narg(__VA_ARGS__), ##__VA_ARGS__)

Definition at line 256 of file strings.h.

Referenced by encrypted_chunk_get(), naked_message_set(), org_key_get(), org_signet_fingerprint(), org_signet_generate(), org_signet_get(), org_signet_verify(), prime_pem_wrap(), signature_full_get(), signature_tree_get(), user_key_get(), user_request_generate(), user_request_get(), user_request_rotation(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_get(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.78.1.20 #define va_narg(...) (__VA_NARG__(_0, ##__VA_ARGS__, __VA_NARG_SEQ_N0) - 1)

Definition at line 259 of file strings.h.

5.78.2 Typedef Documentation**5.78.2.1 typedef void stringer_t**

Definition at line 76 of file strings.h.

5.78.3 Enumeration Type Documentation**5.78.3.1 anonymous enum**

Enumerator:

CONSTANT_T

PLACER_T

NULLER_T

BLOCK_T

MANAGED_T

MAPPED_T

CONTIGUOUS

JOINTED

STACK

HEAP

SECURE

FOREIGNDATA

Definition at line 13 of file strings.h.

5.78.4 Function Documentation**5.78.4.1 struct __attribute__((packed)) [read]**

Definition at line 68 of file strings.h.

References data, and length.

5.78.4.2 `int32_t cmp_mt_mt (multi_t one, multi_t two)`

[multi.c](#) [multi.c](#)

Note:

The types of the two input objects must match, or be a comparison between a managed string and null-terminated string.

Parameters:

- one* the first multi-type object to be compared.
- two* the second multi-type object to be compared.

Returns:

Definition at line 682 of file `multi.c`.

References `multi_t::binary`, `multi_t::dbl`, `multi_t::fl`, `multi_t::i16`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `log_options`, `log_pedantic`, `M_LOG_PEDANTIC`, `M_LOG_STACK_TRACE`, `M_TYPE_BOOLEAN`, `M_TYPE_DOUBLE`, `M_TYPE_FLOAT`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `mt_get_type()`, `multi_t::ns`, `NULLER`, `multi_t::st`, `st_cmp_cs_eq()`, `multi_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, and `multi_t::val`.

Referenced by `tree_cmp()`.

5.78.4.3 `bool_t ident_mt_mt (multi_t one, multi_t two)`

Check to see if the values of two multi-type objects are identical.

Note:

The types of the two input objects must match, or be a comparison between a managed string and null-terminated string.

Parameters:

- one* the first multi-type object to be compared.
- two* the second multi-type object to be compared.

Returns:

true if the two objects have identical values; false if they don't, or on failure.

Definition at line 567 of file `multi.c`.

References `multi_t::binary`, `multi_t::dbl`, `multi_t::fl`, `multi_t::i16`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `log_options`, `log_pedantic`, `M_LOG_PEDANTIC`, `M_LOG_STACK_TRACE`, `M_TYPE_BOOLEAN`, `M_TYPE_DOUBLE`, `M_TYPE_FLOAT`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `mt_get_type()`, `multi_t::ns`, `ns_length_get()`, `PLACER`, `multi_t::st`, `st_cmp_cs_eq()`, `multi_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, and `multi_t::val`.

Referenced by `hashed_bucket_find_key()`, `hashed_delete()`, `linked_delete()`, and `linked_find()`.

5.78.4.4 `multi_t mt_dupe (multi_t multi)`

Duplicate a multi-type object.

Parameters:

- multi* the multi-type object to be examined.

Returns:

a copy of the input object (a deep copy for managed strings and null-terminated strings).

Definition at line 484 of file multi.c.

References multi_t::binary, CONTIGUOUS, multi_t::dbl, multi_t::fl, HEAP, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, MANAGED_T, mt_get_null(), multi_t::ns, ns_dupe(), multi_t::st, st_dupe_opts(), type(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by hashed_bucket_alloc(), linked_record_alloc(), and tree_insert().

5.78.4.5 void mt_free (multi_t multi)

Free the data underlying a multi-type data object (for managed and null-terminated strings).

Parameters:

multi the multi-type object to be freed.

Returns:

This function returns no value.

Definition at line 462 of file multi.c.

References M_TYPE_NULLER, M_TYPE_STRINGER, multi_t::ns, ns_free(), multi_t::st, st_free(), multi_t::type, and multi_t::val.

Referenced by hashed_delete(), hashed_free(), hashed_truncate(), linked_record_free(), tree_delete(), tree_insert(), and tree_truncate().

5.78.4.6 char* mt_get_char (const multi_t * multi)

Get a character pointer to the value of a multi-type object.

Parameters:

multi a pointer to the multi-type object to be examined.

a pointer to the value of the input object, or NULL on failure.

Returns:

null on error

Definition at line 172 of file multi.c.

References multi_t::binary, multi_t::bl, multi_t::dbl, multi_t::fl, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, log_options, M_LOG_INFO, M_LOG_STACK_TRACE, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, multi_t::ns, multi_t::st, st_char_get(), type(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by hashed_bucket().

5.78.4.7 size_t mt_get_length (multi_t multi)

Get the length of the data associated with a multi-type object.

Parameters:

multi the multi-type object to be examined.

Returns:

the length in bytes of the data associated with the input object, or 0 on failure.

Definition at line 249 of file multi.c.

References log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, multi_t::ns, ns_length_get(), multi_t::st, st_length_get(), type(), multi_t::type, and multi_t::val.

Referenced by hashed_bucket().

5.78.4.8 multi_t mt_get_null (void)

Return an empty multi-type object.

Returns:

an empty multi-type object.

Definition at line 17 of file multi.c.

References EMPTY, and multi_t::type.

Referenced by hashed_cursor_key_active(), hashed_cursor_key_next(), inx_cursor_key_active(), inx_cursor_key_next(), linked_cursor_key_active(), linked_cursor_key_next(), linked_record_get_key(), mt_dupe(), tree_cursor_key_next(), and tree_cursor_next().

5.78.4.9 uint64_t mt_get_number (multi_t multi)

Get the value of a numerical multi-type object.

Parameters:

multi the multi-type object to be examined.

Returns:

the numerical value of the input object, or 0 on failure.

Definition at line 116 of file multi.c.

References multi_t::dbl, multi_t::fl, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_EMPTY, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, type(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by hashed_bucket().

5.78.4.10 M_TYPE mt_get_type (multi_t multi)

Get the data type associated with a multi-type object.

Parameters:

multi the multi-type object to be examined.

Returns:

the data type of the input object, or EMPTY on failure.

Definition at line 318 of file multi.c.

References EMPTY, log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_PLACER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, type(), and multi_t::type.

Referenced by cmp_mt_mt(), and ident_mt_mt().

5.78.4.11 bool_t mt_is_empty (multi_t multi)

Determine whether a multi-type object is empty.

Note:

All numeric (including boolean and floating point) types will always return false.

Parameters:

multi the multi-type object to be examined.

Returns:

true if the object is empty (by type or value), or false otherwise.

Definition at line 31 of file multi.c.

References multi_t::bl, log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_EMPTY, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, multi_t::ns, ns_empty(), multi_t::st, st_empty, type(), multi_t::type, and multi_t::val.

Referenced by config_load_cmdline_settings(), config_load_file_settings(), and tree_insert().

5.78.4.12 bool_t mt_is_number (multi_t multi)

Determine whether a multi-type object is a number.

Parameters:

multi the multi-type object to be examined. return true if the input object is any integer, boolean, or floating point; false otherwise.

Definition at line 80 of file multi.c.

References log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_EMPTY, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, type(), and multi_t::type.

Referenced by hashed_bucket().

5.78.4.13 multi_t mt_set_type (multi_t multi, M_TYPE target)

Set the data type of a multi-type data object.

Parameters:

multi the multi-type object to be adjusted.

target the value of the new data type for the input object.

Returns:

a copy of the modified multi-type data object.

Definition at line 391 of file multi.c.

References EMPTY, log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_PLACER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, type(), and multi_t::type.

Referenced by nvp_parse().

5.78.4.14 chr_t* ns_alloc (size_t len)

[nuller.c](#) [nuller.c](#)

Parameters:

len the length of the buffer to be allocated.

Returns:

NULL on failure or if len was 0; a pointer to the newly allocated memory otherwise.

Definition at line 68 of file nuller.c.

References log_pedantic, mm_alloc(), and ns_wipe().

Referenced by deserialize_ns(), mail_message_path(), and ns_append().

5.78.4.15 chr_t* ns_append (chr_t * s, chr_t * append)

Append one string to another and return the result.

Parameters:

s a pointer to a leading null-terminated string to to which the other string will be appended.

append a pointer to a null-terminated string to be appended to the leading string.

Returns:

NULL if a memory allocation failure occurred, or a pointer to the resulting string on success.

Definition at line 211 of file nuller.c.

References log_pedantic, mm_copy(), ns_alloc(), ns_dupe(), ns_free(), and ns_length_get().

Referenced by api_response(), portal_endpoint_error(), and portal_endpoint_response().

5.78.4.16 void ns_cleanup_variadic (ssize_t len, ...)

A checked cleanup function which can be used free a variable number of null-terminated strings.

See also:

[ns_free\(\)](#)

Parameters:

va_list a list of null-terminated strings to be freed.

Returns:

This function returns no value.

Definition at line 146 of file nuller.c.

References ns_free().

5.78.4.17 chr_t* ns_dup (chr_t * s)

Duplicate a null-terminated string.

Parameters:

s the null-terminated string to be duplicated.

Returns:

NULL on failure, or a pointer to a copy of the input string.

Definition at line 167 of file nuller.c.

References length, log_info, log_pedantic, mm_alloc(), mm_copy(), and ns_length_get().

Referenced by cache_set_value(), config_value_set(), mt_dup(), ns_append(), relay_set_value(), servers_set_value(), and xml_get_xpath_ns().

5.78.4.18 bool_t ns_empty (chr_t * s)

Return whether a null-terminated string is empty or not.

Parameters:

s the input as a null-terminated string.

Returns:

true if string is NULL or zero length; false otherwise.

Definition at line 29 of file nuller.c.

References ns_length_get().

Referenced by cache_free(), cache_output_settings(), cache_set_value(), cache_validate(), config_output_value(), config_output_value_generic(), config_value_set(), dkim_signature_create(), lib_load(), mail_mime_get_media_type(), mt_is_empty(), relay_free(), relay_output_settings(), relay_set_value(), relay_validate(), servers_free(), servers_output_settings(), servers_set_value(), servers_validate(), sql_start(), and st_merge_opts().

5.78.4.19 bool_t ns_empty_out (chr_t * s, chr_t ** ptr, size_t * len)

Return whether a null-terminated string is empty or not, while storing its address and length.

Parameters:

s the input as a null-terminated string.

ptr a pointer address to receive a copy of the string's location.

len a pointer to a variable to receive the length of the string, in bytes.

Returns:

true if string is NULL or zero length; false otherwise.

Definition at line 41 of file nuller.c.

References ns_length_get().

5.78.4.20 void ns_free (chr_t * s)

Free a null-terminated string.

Parameters:

s the string to be freed.

Returns:

This function returns no value.

Definition at line 95 of file nuller.c.

References log_pedantic, and mm_free().

Referenced by api_response(), cache_free(), cache_set_value(), config_free(), config_value_set(), ip_subnet_st(), mail_copy_message(), mail_load_message(), mail_message_path(), mail_remove_message(), mail_store_message(), mail_store_message_data(), mt_free(), ns_append(), ns_cleanup_variadic(), portal_endpoint_error(), portal_endpoint_response(), relay_free(), relay_set_value(), servers_free(), and servers_set_value().

5.78.4.21 chr_t* ns_import (void * block, size_t len)

Get a block of memory as a null-terminated string.

Parameters:

block a pointer to the buffer containing the data to be copied.

len the length, in bytes, of the data buffer to be copied.

Returns:

NULL on failure, or a pointer to the newly allocated null-terminated string on success.

Definition at line 192 of file nuller.c.

References log_pedantic, mm_alloc(), and mm_copy().

Referenced by cache_set_value(), config_value_set(), ip_subnet_st(), relay_set_value(), and servers_set_value().

5.78.4.22 size_t ns_length_get (const chr_t * s)

Return the length of a null-terminated string. LOW: The ns_ functions are designed to work with NULL terminated strings. They shouldn't require a length. If the length needs to be provided, then the equivalent mm_ function should be used.

Parameters:

s the input as a null-terminatd string.

Returns:

the length in bytes of the string.

Definition at line 18 of file nuller.c.

Referenced by `api_endpoint_delete_user()`, `api_endpoint_register()`, `api_response()`, `build_version_major()`, `build_version_minor()`, `build_version_patch()`, `con_reverse_lookup()`, `con_write_ns()`, `config_fetch_host_number()`, `config_fetch_settings()`, `deprecated_ecies_key_private_hex()`, `deprecated_ecies_key_public_hex()`, `ecies_key_private_hex()`, `ecies_key_public_hex()`, `file_temp_handle()`, `host_platform()`, `host_version()`, `http_load_file()`, `ident_mt_mt()`, `imap_build_array()`, `imap_fetch_bodystructure()`, `imap_fetch_message()`, `imap_search_messages_date()`, `int16_conv_ns()`, `int32_conv_ns()`, `int64_conv_ns()`, `int8_conv_ns()`, `ip_addr_st()`, `ip_presentation()`, `ip_subnet_st()`, `lib_load_openssl()`, `line_pl_ns()`, `log_rotate()`, `log_start()`, `mail_mime_get_media_type()`, `mail_path_finder()`, `meta_folders_name()`, `mt_get_length()`, `ns_append()`, `ns_dupe()`, `ns_empty()`, `ns_empty_out()`, `ns_length_int()`, `portal_endpoint()`, `portal_endpoint_attachments_add()`, `portal_endpoint_error()`, `portal_endpoint_folders_add()`, `portal_endpoint_folders_rename()`, `portal_endpoint_response()`, `portal_parse_json_str_array()`, `rand_choices()`, `register_data_insert_user()`, `register_print_step1()`, `register_print_step2()`, `servers_network_start()`, `smtp_client_send_helo()`, `spool_path()`, `st_merge_opts()`, `stmt_rebuild()`, `stmt_start()`, `tank_start()`, `teacher_add_cookie()`, `uint16_conv_ns()`, `uint32_conv_ns()`, `uint64_conv_ns()`, `uint8_conv_ns()`, `virus_check()`, `xml_get_xpath_int16()`, `xml_get_xpath_int32()`, `xml_get_xpath_int64()`, `xml_get_xpath_int8()`, `xml_get_xpath_st()`, `xml_get_xpath_uint16()`, `xml_get_xpath_uint32()`, `xml_get_xpath_uint64()`, and `xml_get_xpath_uint8()`.

5.78.4.23 int ns_length_int (chr_t * s)

Return the length of a null-terminated string as an int, capped at INT_MAX.

Parameters:

s the null-terminated string to be evaluated.

Returns:

the length of the string.

Definition at line 51 of file nuller.c.

References `log_pedantic`, and `ns_length_get()`.

Referenced by `lib_load_bzip()`.

5.78.4.24 bool_t ns_populated_variadic (ssize_t len, ...)

A simple method for checking multiple NULL terminated strings to ensure all of the provided pointers lead to a buffer with at least one non-NULL character.

Parameters:

len the number of pointers being passed in.

va_list the list of NULL terminated string pointers to validate.

Returns:

true if none of the pointers are NULL, and all point to at least one non-NULL byte, otherwise false.

Definition at line 115 of file nuller.c.

5.78.4.25 void ns_wipe (chr_t * s, size_t len)

Zero out a null-terminated string.

Parameters:

s the string to be wiped.

len the number of bytes to be wiped at the start of the string.

Returns:

This function returns no parameters.

Definition at line 246 of file nuller.c.

References `log_pedantic`, and `mm_set()`.

Referenced by `deprecated_ecies_key_private_hex()`, `ecies_key_private_hex()`, and `ns_alloc()`.

5.78.4.26 `chr_t* pl_char_get (placer_t place)`

[shortcuts.c shortcuts.c](#)

Parameters:

place the input placer.

Returns:

NULL on failure or a character pointer to the block of data associated with the specified placer on success.

Definition at line 54 of file shortcuts.c.

References `st_char_get()`.

Referenced by `cache_config()`, `con_write_pl()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_public()`, `dmtpl_process()`, `ecies_key_private()`, `ecies_key_public()`, `imap_fetch_body_header()`, `imap_narrow_messages()`, `imap_search_messages_range()`, `ip_subnet_st()`, `lib_load_openssl()`, `line_pl_pl()`, `molten_parse()`, `nvp_parse()`, `pl_get_embraced()`, `pl_shrink_before_characters()`, `pl_skip_characters()`, `pl_skip_to_characters()`, `pl_starts_with_char()`, `pl_trim()`, `pl_trim_end()`, `pl_trim_start()`, `pop_process()`, `portal_message_body()`, `portal_upload()`, `relay_config()`, `servers_config()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_client_send_data()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, `smtp_parse_rcpt_to()`, `smtp_process()`, and `str_tok_get_bl()`.

5.78.4.27 `placer_t pl_clone (placer_t place)`

Definition at line 30 of file shortcuts.c.

References `pl_init()`.

Referenced by `prime_pem_unwrap()`.

5.78.4.28 `void* pl_data_get (placer_t place)`

Get a pointer to the data referenced by a placer.

Parameters:

place the input placer.

Returns:

NULL on failure or a pointer to the block of data associated with the specified placer on success.

Definition at line 44 of file shortcuts.c.

References `st_data_get()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_public()`, `ecies_key_private()`, `ecies_key_public()`, `ephemeral_chunk_set()`, `imap_build_array()`, `imap_build_array_isliteral()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_parse_address_part()`, `nvp_parse()`, `org_signet_set()`, `prime_pem_unwrap()`, `smtp_check_filters()`, `tok_get_pl()`, `uint64_conv_pl()`, `user_request_set()`, and `user_signet_set()`.

5.78.4.29 bool_t pl_empty (placer_t *place*)

Determine whether or not the specified placer is empty.

Parameters:

place the input placer.

Returns:

true if the placer is empty or zero-length, or false otherwise.

Definition at line 84 of file shortcuts.c.

References st_empty.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), cache_config(), client_read_line(), con_read_line(), dmtip_process(), http_data_value_parse(), http_process(), imap_build_array(), imap_build_array_isliteral(), imap_fetch_body(), imap_fetch_body_header(), imap_narrow_messages(), imap_parse_address(), imap_parse_literal(), imap_process(), imap_search_messages_date(), imap_search_messages_header(), imap_search_messages_range(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_mime_child(), mail_mime_count(), mail_mod_subject(), molten_parse(), multipart_get_boundary(), nvp_parse(), pl_get_embraced(), pl_inc(), pl_shrink_before_characters(), pl_skip_characters(), pl_skip_to_characters(), pl_starts_with_char(), pop_process(), portal_upload(), prime_pem_unwrap(), relay_config(), servers_config(), slots_key(), smtp_parse_helo_domain(), smtp_parse_mail_from_path(), smtp_parse_rcpt_to(), and smtp_process().

5.78.4.30 bool_t pl_inc (placer_t * *place*, bool_t *more*)

Advance the placer one character forward beyond an expected character.

Parameters:

place the input placer.

more if true, the placer must contain more data, and vice versa.

Returns:

true if more was true and the placer contains more data, or if more was false and the placer ended; false otherwise.

Definition at line 114 of file shortcuts.c.

References pl_empty().

5.78.4.31 placer_t pl_init (void * *data*, size_t *len*)

Return a placer wrapping a data buffer of given size.

Parameters:

data a pointer to the data to be wrapped.

len the length, in bytes, of the data.

Returns:

a placer pointing to the specified data.

Definition at line 25 of file shortcuts.c.

References FOREIGNDATA, JOINTED, PLACER_T, placer_t, and STACK.

Referenced by `aes_cipher_key()`, `aes_tag_shard()`, `aes_vector_shard()`, `api_endpoint_register()`, `bracket_extract_pl()`, `chunk_header_read()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `ecies_decrypt()`, `ecies_encrypt()`, `ephemeral_chunk_set()`, `get_header_value_noopt()`, `http_parse_context()`, `http_parse_origin()`, `http_parse_pairs()`, `imap_build_array()`, `imap_fetch_body()`, `imap_fetch_body_portion()`, `imap_fetch_bodystructure()`, `imap_fetch_envelope()`, `imap_parse_address_breaker()`, `imap_search_messages_date()`, `imap_search_messages_header()`, `line_pl_bl()`, `mail_get_chunk()`, `mail_header_pop()`, `mail_mime_child()`, `mail_mime_header()`, `mail_mime_part()`, `mail_mime_update()`, `mail_store_header()`, `naked_message_get()`, `naked_message_set()`, `pl_clone()`, `pl_trim()`, `pl_trim_end()`, `pl_trim_start()`, `prime_pem_unwrap()`, `prime_reader_open()`, `prime_reader_payload()`, `signature_full_verify()`, `signature_tree_verify()`, `slots_alloc()`, `slots_get()`, `smtp_check_filters()`, `smtp_client_send_helo()`, `str_tok_get_bl()`, `str_tok_get_count_bl()`, `tok_get_ns()`, and `tok_pop()`.

5.78.4.32 `size_t pl_length_get (placer_t place)`

Get the length, in bytes, of a placer.

Parameters:

place the input placer.

Returns:

the size, in bytes, of the specified placer.

Definition at line 74 of file shortcuts.c.

References `st_length_get()`.

Referenced by `cache_config()`, `chunk_buffer_read()`, `chunk_header_read()`, `client_read()`, `client_read_line()`, `con_read()`, `con_read_line()`, `con_write_pl()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_public()`, `dmtp_process()`, `ecies_key_private()`, `ecies_key_public()`, `imap_build_array()`, `imap_build_array_isliteral()`, `imap_fetch_body()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_parse_address_part()`, `imap_parse_literal()`, `ip_subnet_st()`, `line_pl_pl()`, `molten_parse()`, `nvp_parse()`, `org_signet_set()`, `pl_shrink_before_characters()`, `pl_trim()`, `pl_trim_end()`, `pl_trim_start()`, `pop_process()`, `portal_message_body()`, `portal_upload()`, `prime_pem_unwrap()`, `relay_config()`, `servers_config()`, `slots_buffer()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_check_filters()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, `smtp_parse_rcpt_to()`, `smtp_process()`, `str_tok_get_bl()`, `str_tok_get_count_bl()`, `tok_get_pl()`, `uint64_conv_pl()`, `user_request_set()`, and `user_signet_set()`.

5.78.4.33 `int_t pl_length_int (placer_t place)`

Get the length, in bytes, of a placer as an integer.

Parameters:

place the input placer.

Returns:

the size, in bytes, of the specified placer.

Definition at line 64 of file shortcuts.c.

References `st_length_int()`.

Referenced by `lib_load_openssl()`, `smtp_client_send_data()`, `smtp_parse_helo_domain()`, `smtp_parse_mail_from_path()`, and `smtp_parse_rcpt_to()`.

5.78.4.34 `placer_t pl_null (void)`

Return a zero-length placer pointing to NULL data.

Returns:

a zero-length placer pointing to NULL data.

Definition at line 14 of file shortcuts.c.

References FOREIGNDATA, JOINTED, PLACER_T, placer_t, and STACK.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), aes_cipher_key(), aes_tag_shard(), aes_vector_shard(), bracket_extract_pl(), build_version_major(), build_version_minor(), build_version_patch(), client_read(), client_read_line(), con_init_network_buffer(), con_read(), con_read_line(), get_header_opt(), imap_fetch_body(), imap_fetch_body_portion(), imap_fetch_envelope(), imap_narrow_messages(), imap_parse_address_breaker(), imap_parse_literal(), imap_search_messages_date(), imap_search_messages_header(), imap_search_messages_range(), imap_starttls(), line_pl_bl(), mail_get_chunk(), mail_header_pop(), mail_mime_child(), mail_modify_part(), naked_message_get(), naked_message_set(), nvp_parse(), part_decrypt(), pl_trim(), pl_trim_end(), pl_trim_start(), pop_starttls(), prime_pem_unwrap(), prime_reader_payload(), slots_buffer(), smtp_check_filters(), smtp_client_send_helo(), smtp_rcpt_to(), smtp_starttls(), str_tok_get_bl(), tok_get_ns(), and tok_pop().

5.78.4.35 placer_t pl_set (placer_t *place*, placer_t *set*)

Definition at line 34 of file shortcuts.c.

References placer_t.

Referenced by mail_mime_split().

5.78.4.36 bool_t pl_starts_with_char (placer_t *place*, chr_t *c*)

Determine if a placer begins with a specified character.

Parameters:

- place* the input placer.
- c* the character to be compared with the first byte of the placer's data.

Returns:

true if the placer begins with the given character or false otherwise.

Definition at line 95 of file shortcuts.c.

References pl_char_get(), and pl_empty().

Referenced by nvp_parse(), prime_pem_unwrap(), and smtp_client_send_data().

5.78.4.37 stringer_t* st_alloc (size_t *len*)

Allocate a managed string with a specified options mask.

See also:

[st_alloc_opts\(\)](#)

Parameters:

- len* the length, in bytes, of the managed string to be allocated.

Returns:

NULL on failure or a pointer to the newly allocated managed string on success.

Definition at line 631 of file allocation.c.

References CONTIGUOUS, HEAP, MANAGED_T, and st_alloc_opts().

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `auth_legacy()`, `auth_sanitizer_address()`, `base64_decode()`, `base64_decode_mod()`, `base64_encode()`, `base64_encode_mod()`, `base64_encode_wrap()`, `client_connect()`, `con_init_network_buffer()`, `decompress_block_lzo()`, `decompress_bzip()`, `decompress_lzo()`, `decompress_zlib()`, `deprecated_hmac_digest()`, `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `deserialize_st()`, `ed25519_private_get()`, `ed25519_public_get()`, `ed25519_sign()`, `encrypted_chunk_set()`, `file_load()`, `hash_digest()`, `hex_decode_st()`, `hex_encode_st()`, `hex_encode_st_debug()`, `hmac_digest()`, `http_load_file()`, `imap_build_array()`, `imap_parse_address_part()`, `imap_parse_literal()`, `imap_parse_qstring()`, `ip_presentation()`, `ip_reversed()`, `ip_standard()`, `ip_subnet()`, `mail_add_required_headers()`, `mail_build_signature()`, `mail_extract_address()`, `mail_extract_tag()`, `mail_header_fetch_cleaned()`, `mail_load_message()`, `mail_mime_generate_boundary()`, `org_key_get()`, `org_signet_get()`, `portal_smtp_merge_headers()`, `qp_decode()`, `rand_choices()`, `sanity_check()`, `secp256k1_compute_kek()`, `secp256k1_private_get()`, `secp256k1_public_get()`, `serialize_int16()`, `serialize_int32()`, `serialize_int64()`, `serialize_uint16()`, `serialize_uint32()`, `serialize_uint64()`, `signature_full_get()`, `signature_tree_get()`, `st_bitwise()`, `st_not()`, `st_output()`, `st_replace()`, `stacie_create_nonce()`, `stacie_create_salt()`, `stacie_create_shard()`, `stacie_decrypt()`, `stacie_encrypt()`, `symmetric_decrypt()`, `symmetric_encrypt()`, `uint16_put_no()`, `uint24_put_no()`, `uint32_put_no()`, `url_decode()`, `user_key_get()`, `user_request_get()`, `user_signet_get()`, `zbase32_decode()`, and `zbase32_encode()`.

5.78.4.38 `stringer_t* st_alloc_opts (uint32_t opts, size_t len)`

Allocate a managed string with a specified options mask.

See also:

`st_valid_options()`

Note:

All requested allocation masks should conform to the validation imposed by `st_valid_opts()`. The supported types are: `placer`, `nuller`, `block`, `managed`, and `mapped`. The following logic is applied to requested string allocation options: 1. Any allocation options specified for strings to be allocated on the stack are IGNORED. 2. All jointed strings allocate memory for the header and data separately and then link them EXCEPT: Jointed placers only allocate space for a header. Jointed mapped strings allocate the data with an aligned `mmap()` operation. 3. Contiguous strings allocate space for the header and data together in a single contiguous block. 4. After allocation, the length field is set for block strings. 5. After allocation, the available field is set for managed strings. 6. Placers can only be jointed; mapped strings can only be contiguous.

Parameters:

opts the allocation options mask to be used by the newly allocated string.

len the length, in bytes, of the managed string to be allocated.

Returns:

NULL on failure or a pointer to the newly allocated managed string on success.

Definition at line 498 of file `allocation.c`.

References `align()`, `block_t`, `BLOCK_T`, `CONTIGUOUS`, `HEAP`, `JOINTED`, `log_pedantic`, `magma`, `MAGMA_SPOOL_DATA`, `managed_t`, `MANAGED_T`, `mapped_t`, `MAPPED_T`, `MEMORYBUF`, `mm_alloc()`, `mm_free()`, `mm_sec_alloc()`, `mm_sec_free()`, `mm_set()`, `nuller_t`, `NULLER_T`, `magma_t::page_length`, `placer_t`, `PLACER_T`, `SECURE`, `spool_mktemp()`, `st_info_opts()`, `st_valid_opts()`, and `STACK`.

Referenced by `auth_legacy()`, `base64_decode_opts()`, `base64_encode_opts()`, `dkim_signature_create()`, `ephemeral_chunk_set()`, `hex_decode_opts()`, `hex_encode_opts()`, `http_body()`, `mail_add_forward_headers()`, `mail_message_cleanup()`, `mail_mime_split()`, `naked_message_set()`, `part_decrypt()`, `pop_pass_parse()`, `qp_encode()`, `signature_tree_get()`, `signature_tree_verify()`, `smtp_check_greylist()`, `smtp_data_read()`, `st_alloc()`, `st_append_opts()`, `st_append_out()`, `st_dupe_opts()`, `st_import_opts()`, `st_merge_opts()`, `st_nullify()`, `st_vaprint_opts()`, `stacie_derive_key()`, `stacie_derive_seed()`, `stacie_derive_token()`, `stacie_realm_key()`, and `url_encode()`.

5.78.4.39 `stringer_t* st_append_opts (size_t align, stringer_t * s, stringer_t * append)`

Append one managed string to another, with aligned memory boundaries.

Parameters:

align an alignment value to be used for managed string (re)allocation. This only works for powers of 2.

s the managed string to be extended, which will be allocated for the caller if passed as NULL.

append the managed string to be appended to *s*.

Returns:

NULL on failure, or a pointer to the appended result on success.

Definition at line 407 of file allocation.c.

References HEAP, JOINTED, log_pedantic, MANAGED_T, mm_copy(), st_alloc_opts(), st_avail_get(), st_data_get(), st_length_get(), st_length_set(), st_realloc(), and st_valid_append().

Referenced by http_body(), imap_id(), imap_search(), imap_status(), and mail_add_forward_headers().

5.78.4.40 int_t st_append_out (size_t align, stringer_t **s, stringer_t *append)

Definition at line 439 of file allocation.c.

References HEAP, JOINTED, log_pedantic, MANAGED_T, mm_copy(), st_alloc_opts(), st_avail_get(), st_data_get(), st_free(), st_length_get(), st_length_set(), st_realloc(), and st_valid_destination().

Referenced by encrypted_chunk_set().

5.78.4.41 stringer_t* st_aprint (chr_t *format, ...)

[print.c](#)

5.78.4.42 stringer_t stringer_t* st_aprint_opts (uint32_t opts, chr_t *format, ...)

5.78.4.43 size_t st_avail_get (stringer_t *s)

Return the total data buffer size allocated for a managed string.

Parameters:

s the input managed string.

Returns:

0 on failure, or the total buffer size in bytes on success.

Definition at line 146 of file length.c.

References log_pedantic, managed_t, MANAGED_T, mapped_t, MAPPED_T, MEMORYBUF, st_info_opts(), st_length_get(), and st_valid_opts().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), client_read(), client_read_line(), con_read(), con_read_line(), deprecated_hmac_digest(), ed25519_private_get(), ed25519_public_get(), ed25519_sign(), file_load(), file_read(), hash_digest(), hex_decode_st(), hex_encode_st(), hmac_digest(), host_platform(), host_version(), imap_parse_address_put(), ip_presentation(), ip_reversed(), ip_standard(), ip_subnet(), mail_add_required_headers(), meta_data_fetch_shard(), org_key_get(), org_signet_get(), rand_choices(), rand_write(), sanity_check(), secp256k1_compute_kek(), secp256k1_private_get(), secp256k1_public_get(), serialize_int16(), serialize_int32(), serialize_int64(), serialize_ns(), serialize_st(), serialize_uint16(), serialize_uint32(), serialize_uint64(), st_append_opts(), st_append_out(), st_bitwise(), st_copy_in(), st_dupe_opts(), st_not(), st_output(), st_set(), st_trim(), st_vsprint(), st_wipe(), st_write_variadic(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), time_print_gmt(), time_print_local(), user_key_get(), user_request_get(), and user_signet_get().

5.78.4.44 `size_t st_avail_set (stringer_t * s, size_t avail)`

Set the total data buffer size allocated for a managed string.

Parameters:

s the input managed string.

avail the new buffer size for the managed string.

Returns:

0 on failure, or the managed string's new buffer size in bytes on success.

Definition at line 183 of file length.c.

References `log_pedantic`, `managed_t`, `MANAGED_T`, `mapped_t`, `MAPPED_T`, `MEMORYBUF`, `st_info_opts()`, and `st_valid_avail()`.

5.78.4.45 `chr_t* st_char_get (stringer_t * s)`

data.c data.c

See also:

[st_data_get\(\)](#)

Parameters:

s the input managed string.

Returns:

NULL on failure or for an improperly constructed string; otherwise, a pointer to the string's data.

Definition at line 196 of file data.c.

References `st_data_get()`.

Referenced by `api_endpoint_auth()`, `api_endpoint_change_password()`, `api_endpoint_delete_user()`, `auth_data_fetch()`, `auth_data_update_legacy()`, `auth_login()`, `auth_response()`, `auth_sanitise_address()`, `auth_sanitise_username()`, `cache_add()`, `cache_append()`, `cache_config()`, `cache_decrement()`, `cache_delete()`, `cache_get()`, `cache_get_u64()`, `cache_increment()`, `cache_output_settings()`, `cache_set()`, `cache_set_u64()`, `cache_set_value()`, `cache_silent_add()`, `cipher_name()`, `client_read()`, `client_read_line()`, `client_write()`, `con_read()`, `con_read_line()`, `con_write_st()`, `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_file_settings()`, `config_output_value()`, `config_output_value_generic()`, `config_validate_settings()`, `config_value_set()`, `contact_abuse_checks()`, `contact_abuse_increment_history()`, `contact_business_valid_email()`, `contact_detail_delete()`, `contact_detail_upsert()`, `contact_insert()`, `contact_print_form()`, `contact_update()`, `deserialize_int16()`, `deserialize_int32()`, `deserialize_int64()`, `deserialize_ns()`, `deserialize_st()`, `deserialize_uint16()`, `deserialize_uint32()`, `deserialize_uint64()`, `digest_name()`, `dkim_start()`, `dmtip_ehlo()`, `dmtip_helo()`, `dmtip_init()`, `double_conv()`, `dspam_check()`, `dspam_train()`, `file_temp_handle()`, `float_conv()`, `folder_count()`, `folder_exists()`, `get_header_value_noopt()`, `hex_encode_st_debug()`, `http_body()`, `http_content_load_directory()`, `http_data_header_parse()`, `http_data_value_decode()`, `http_data_value_parse()`, `http_load_file()`, `http_page_get()`, `http_parse_context()`, `http_parse_header()`, `http_parse_method()`, `http_parse_pairs()`, `http_print_301()`, `http_response_allow_cross()`, `http_response_cookie()`, `http_response_header()`, `http_response_options()`, `imap_append()`, `imap_append_message()`, `imap_build_array()`, `imap_capability()`, `imap_check()`, `imap_close()`, `imap_command_log_safe()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_fetch_body_portion()`, `imap_fetch_bodystructure()`, `imap_fetch_envelope()`, `imap_fetch_parse_partial()`, `imap_fetch_return_header()`, `imap_folder_compare()`, `imap_id()`, `imap_idle()`, `imap_init()`, `imap_invalid()`, `imap_list()`, `imap_login()`, `imap_logout()`, `imap_lsub()`, `imap_narrow_messages()`, `imap_noop()`, `imap_parse_address_breaker()`, `imap_parse_address_part()`, `imap_parse_address_put()`, `imap_parse_literal()`, `imap_parse_qstring()`, `imap_process()`, `imap_rename()`, `imap_search()`, `imap_search_messages_date()`, `imap_search_messages_header()`, `imap_select()`, `imap_starttls()`, `imap_status()`, `imap_store()`, `imap_subscribe()`, `imap_unsubscribe()`, `ip_presentation()`, `ip_reversed()`, `ip_standard()`, `json_api_dispatch()`, `lib_load()`, `line_pl_st()`, `lock_get()`, `lock_release()`, `magma_folder_insert()`, `magma_folder_rename()`, `mail_add_forward_headers()`, `mail_add_outbound_headers()`, `mail_add_required_headers()`, `mail_build_signature()`, `mail_create_directory()`, `mail_db_insert_duplicate_message()`, `mail_db_`

insert_message(), mail_discover_encoding(), mail_discover_insertion_point(), mail_discover_type(), mail_extract_address(), mail_extract_tag(), mail_get_boundary(), mail_get_chunk(), mail_header_end(), mail_header_fetch_all(), mail_header_fetch_cleaned(), mail_header_pop(), mail_headers(), mail_insert_chunk_base64(), mail_insert_chunk_text(), mail_load_header(), mail_load_message(), mail_load_message_top(), mail_message(), mail_message_cleanup(), mail_message_path(), mail_mime_boundary(), mail_mime_child(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_count(), mail_mime_encoding(), mail_mime_generate_boundary(), mail_mime_header(), mail_mime_part(), mail_mime_type(), mail_mime_type_group(), mail_mime_type_parameters_key(), mail_mime_type_parameters_value(), mail_mime_type_sub(), mail_mime_update(), mail_mod_subject(), mail_modify_part(), mail_signature_add(), meta_crypto_keys_create(), meta_data_delete_tag(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_shard(), meta_data_insert_folder(), meta_data_insert_keys(), meta_data_insert_shard(), meta_data_insert_tag(), meta_data_update_folder_name(), meta_update_keys(), meta_update_realms(), mt_get_char(), net_trigger(), pl_char_get(), pop_pass(), pop_retr(), pop_top(), portal_config_collection(), portal_config_entry(), portal_contact_details(), portal_endpoint(), portal_endpoint_alert_acknowledge(), portal_endpoint_alert_list(), portal_endpoint_aliases(), portal_endpoint_attachments_add(), portal_endpoint_attachments_remove(), portal_endpoint_auth(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_edit(), portal_endpoint_contacts_list(), portal_endpoint_contacts_load(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), portal_endpoint_folders_add(), portal_endpoint_folders_list(), portal_endpoint_folders_remove(), portal_endpoint_folders_rename(), portal_endpoint_folders_tags(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_load(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_endpoint_messages_tags(), portal_message_attachments(), portal_message_body(), portal_message_header(), portal_message_tags_array(), portal_meta(), portal_settings_changepass(), portal_settings_identity(), portal_upload(), portal_validate_request(), prime_start(), process_kill(), protocol_secure(), register_abuse_check_blocklist(), register_abuse_checks(), register_abuse_increment_history(), register_business_step1(), register_business_validate_password(), register_business_validate_username(), register_captcha_generate(), register_data_check_username(), register_data_insert_user(), register_print_captcha(), register_session_cache(), register_session_get(), relay_config(), relay_output_settings(), relay_set_value(), secp256k1_public_set(), serial_get(), serial_increment(), serial_reset(), serialize_int16(), serialize_int32(), serialize_int64(), serialize_ns(), serialize_st(), serialize_uint16(), serialize_uint32(), serialize_uint64(), servers_config(), servers_output_settings(), servers_set_value(), sess_get(), smtp_add_bypass_entry(), smtp_auth_login(), smtp_auth_plain(), smtp_bounce(), smtp_check_authorized_from(), smtp_check_filters(), smtp_check_greylist(), smtp_check_rbl(), smtp_check_receive_quota(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), smtp_client_send_nullfrom(), smtp_client_send_rcptto(), smtp_data_finish(), smtp_data_outbound(), smtp_data_read(), smtp_ehlo(), smtp_fetch_authorization(), smtp_fetch_inbound(), smtp_helo(), smtp_init(), smtp_insert_spamsig(), smtp_rcpt_to(), smtp_reply(), smtp_rollout(), smtp_update_receive_stats(), spf_check(), spool_check(), spool_cleanup(), spool_mktemp(), spool_start(), spool_stop(), sql_query_conn(), st_info_opts(), st_merge_opts(), st_replace(), st_trim(), stacie_derive_key(), stacie_derive_token(), teacher_add_cookie(), teacher_print_message(), teacher_process(), time_print_gmt(), time_print_local(), tok_pop_init_st(), user_config_delete(), user_config_upsert(), virus_engine_create(), and virus_start().

5.78.4.46 void st_cleanup_variadic (ssize_t len, ...)

A checked cleanup function which can be used free a variable number managed strings.

See also:

[st_free\(\)](#)

Parameters:

va_list a list of managed strings to be freed.

Returns:

This function returns no value.

Definition at line 96 of file allocation.c.

References [st_free\(\)](#).

5.78.4.47 stringer_t* st_copy_in (stringer_t * s, void * buf, size_t len)

Copy data into a managed string.

Parameters:

- s* the managed string to store the copied contents of the data.
- buf* a pointer to the buffer containing the data to be copied.
- len* the length of the data to be copied.

Returns:

NULL on failure, or a pointer to the managed string on success.

Definition at line 279 of file allocation.c.

References log_pedantic, mm_copy(), st_avail_get(), st_data_get(), and st_length_set().

Referenced by pop_pass_parse().

5.78.4.48 void* st_data_get (stringer_t * s)

Retrieve the data associated with a managed string.

Parameters:

- s* the input managed string.

Returns:

NULL on failure or for an improperly constructed string; otherwise, a pointer to the string's data.

Definition at line 149 of file data.c.

References block_t, BLOCK_T, constant_t, CONSTANT_T, log_pedantic, managed_t, MANAGED_T, mapped_t, MAPPED_T, MEMORY_BUF, nuller_t, NULLER_T, placer_t, PLACER_T, st_info_opts(), and st_valid_opts().

Referenced by _serialize_ec_pubkey(), aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), aes_cipher_key(), aes_tag_shard(), aes_vector_shard(), alert_alloc(), alias_alloc(), auth_response(), base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), chunk_header_read(), chunk_header_write(), client_read(), client_read_line(), compress_bzip(), compress_import(), compress_lzo(), compress_zlib(), con_read(), con_read_line(), contact_alloc(), contact_detail_alloc(), contact_name(), decompress_block_lzo(), decompress_bzip(), decompress_lzo(), decompress_zlib(), deprecated_hmac_digest(), deprecated_scramble_encrypt(), deprecated_scramble_import(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), deprecated_symmetric_vector(), dkim_signature_create(), dkim_signature_verify(), domain_alloc(), ed25519_private_get(), ed25519_private_set(), ed25519_public_get(), ed25519_public_set(), ed25519_sign(), ed25519_verify(), encrypted_chunk_get(), encrypted_chunk_set(), ephemeral_chunk_set(), file_load(), file_read(), hash_digest(), hex_decode_st(), hex_encode_st(), hmac_digest(), http_body(), http_parse_origin(), imap_command_parser(), imap_fetch_bodystucture(), int16_conv_st(), int32_conv_st(), int64_conv_st(), int8_conv_st(), magma_folder_alloc(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_discover_insertion_point(), mail_extract_address(), mail_get_boundary(), mail_insert_chunk_base64(), mail_mime_header(), mail_mime_part(), mail_mod_subject(), mail_modify_part(), mail_store_message_data(), message_alloc(), meta_data_fetch_shard(), meta_data_insert_shard(), meta_folder_stats_tag_alloc(), naked_message_get(), naked_message_set(), pl_data_get(), pop_num_parse(), pop_pass_parse(), pop_top_parse(), pop_user_parse(), prime_field_write(), prime_header_write(), prime_pem_unwrap(), prime_pem_wrap(), prime_reader_open(), prime_unpack(), qp_decode(), qp_encode(), rand_choices(), rand_write(), register_data_insert_user(), sanity_check(), scramble_encrypt(), scramble_import(), secp256k1_compute_kek(), secp256k1_private_get(), secp256k1_private_set(), secp256k1_public_get(), secp256k1_public_set(), sess_get(), signature_full_verify(), signature_tree_verify(), slots_get(), smtp_check_greylist(), smtp_data_finish(), smtp_data_read(), sql_insert_conn(), sql_num_rows_conn(), sql_query_conn(), sql_query_res_conn(), sql_write_conn(), st_append_opts(), st_append_out(), st_bitwise(), st_char_get(), st_copy_in(), st_dupe_opts(), st_empty_out(), st_empty_variadic(), st_import_opts(), st_merge_opts(), st_not(), st_nullify(), st_populated_variadic(), st_replace(), st_set(), st_uchar_get(), st_vaprint_opts(), st_vsprint(), st_wipe(), st_write_variadic(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_encrypt(), stacie_realm_cipher(), stacie_realm_tag(), stacie_realm_vector(), symmetric_decrypt(), symmetric_encrypt(), symmetric_vector(), tank_store(), tcp_status(), tok_get_count_st(), tok_get_st(), uint16_conv_st(), uint16_get_no(), uint16_put_no(), uint24_get_no(), uint24_put_no(), uint32_conv_st(), uint32_get_no(), uint32_put_no(), uint64_conv_st(), uint8_conv_st(), url_decode(), url_encode(), user_config_entry_alloc(), virus_check(), xml_encode(), zbase32_decode(), and zbase32_encode().

5.78.4.49 void st_data_set (stringer_t * s, void * data)

Set the underlying data of a jointed managed string.

Parameters:

- s* the managed string to be adjusted.
- data* the data buffer to be attached to the input managed string.

Note:

The underlying data of *s* will be released, unless it contains foreign data.

Returns:

This function does not return a value.

Definition at line 92 of file data.c.

References block_t, BLOCK_T, FOREIGNDATA, JOINTED, log_pedantic, managed_t, MANAGED_T, MAPPED_T, MEMORYBUF, mm_free(), mm_sec_free(), mm_sec_secured(), nuller_t, NULLER_T, placer_t, PLACER_T, SECURE, st_info_opts(), and st_valid_jointed().

Referenced by contact_name(), ephemeral_chunk_set(), smtp_data_finish(), smtp_data_read(), and st_dupe_opts().

5.78.4.50 stringer_t* st_dupe (stringer_t * s)

Duplicate a managed string.

See also:

[st_dupe_opts\(\)](#)

Note:

The allocation options of the duplicated string will be the same as that of the source string.

Parameters:

- s* the managed string to be duplicated.

Returns:

NULL on failure, or a copy of the input managed string on success.

Definition at line 393 of file allocation.c.

References st_dupe_opts().

Referenced by ar_dupe(), auth_login(), auth_response(), contact_folder_rename(), file_temp_handle(), http_load_file(), http_print_301(), imap_folder_create(), imap_folder_rename(), imap_login(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_cache_get(), pop_pass(), prime_get(), prime_key_encrypt(), prime_pem_wrap(), register_business_step1(), register_business_step2(), register_print_message(), register_print_step1(), register_print_step2(), register_print_step3(), register_session_get(), smtp_add_recipient(), smtp_data_outbound(), smtp_fetch_inbound(), teacher_process(), and user_request_sign().

5.78.4.51 stringer_t* st_dupe_opts (uint32_t opts, stringer_t * s)

Create a duplicate copy of a managed string with a specified set of allocation options.

See also:

[st_valid_opts\(\)](#)

Note:

Both the source and destination managed strings must have option values that pass `st_valid_opts()`. The following procedure occurs when creating the duplicate managed string: If the destination is a placer, 0 bytes are allocated for the duplicate's underlying data. If the destination is a constant, nuller, or block, its underlying data buffer will be of equal size to the source's data length. If the destination is managed or mapped, and so is the source, its underlying data buffer will be of equal size to the source's available size; but if the source is neither, the underlying data buffer of the destination managed or mapped string will equal the size of the source's data length. A deep copy will be made of the source data to the destination if the destination is NOT a placer.

Parameters:

- opts* the allocation options mask to be used for the newly created managed string.
- s* the target managed string to be duplicated.

Returns:

NULL on failure or a pointer to a copy of the duplicated managed string on success.

Definition at line 319 of file allocation.c.

References BLOCK_T, CONSTANT_T, log_pedantic, MANAGED_T, MAPPED_T, MEMORYBUF, mm_copy(), NULLER_T, PLACER_T, st_alloc_opts(), st_avail_get(), st_data_get(), st_data_set(), st_info_opts(), st_length_get(), st_length_set(), st_valid_opts(), and st_valid_tracked().

Referenced by args_parse(), cache_set_value(), config_value_set(), file_temp_handle(), http_body(), http_data_value_parse(), http_load_file(), http_parse_header(), http_parse_method(), http_response_connection(), imap_fetch_response_add(), imap_folder_create(), imap_folder_rename(), magma_folder_name(), mail_add_forward_headers(), mail_cache_set(), mt_dupe(), portal_message_body(), relay_set_value(), servers_set_value(), sess_create(), smtp_client_send_data(), smtp_fetch_inbound(), smtp_forward_message(), smtp_relay_message(), st_dupe(), stacie_realm_cipher(), stacie_realm_tag(), and stacie_realm_vector().

5.78.4.52 bool_t st_empty_out (stringer_t * s, uchr_t ** ptr, size_t * len)

Determine whether the specified managed string is empty or not, and store its underlying data and data length.

Parameters:

- s* the input managed string.
- ptr* a pointer to a null-terminated string that will receive the address of the managed string's underlying data.
- len* a pointer to store the size of the managed string's underlying data buffer.

Returns:

true if string is NULL or uninitialized or empty; false otherwise.

Definition at line 76 of file data.c.

References st_data_get(), and st_length_get().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), auth_sanitise_address(), base64_decode(), base64_decode_mod(), base64_decode_opts(), base64_encode(), base64_encode_mod(), base64_encode_opts(), base64_encode_wrap(), chunk_buffer_size(), chunk_header_size(), chunk_header_type(), client_write(), contact_name(), contact_validate_detail(), contact_validate_name(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), ecies_decrypt(), ecies_encrypt(), ephemeral_chunk_set(), hex_count_st(), hex_decode_st(), hex_encode_st(), hex_valid_st(), http_parse_origin(), imap_command_log_safe(), imap_count_folder_levels(), imap_fetch_body_portion(), imap_valid_folder_name(), imap_valid_sequence(), lower_st(), mail_count_received(), mail_extract_address(), mail_mime_encode_part(), naked_message_get(), part_decrypt(), part_encrypt(), prime_field_write(), prime_header_read(), qp_decode(), qp_encode(), smtp_parse_auth(), st_cmp_ci_ends(), st_cmp_ci_eq(), st_cmp_ci_starts(), st_cmp_cs_ends(), st_cmp_cs_eq(), st_cmp_cs_starts(), st_replace(), st_search_chr(), st_search_ci(), st_search_cs(), st_swap(), st_write_variadic(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_encrypt(), stacie_realm_key(), symmetric_decrypt(), symmetric_encrypt(), upper_st(), url_decode(), url_encode(), url_valid_st(), utf8_length_st(), utf8_valid_st(), zbase32_decode(), and zbase32_encode().

5.78.4.53 bool_t st_empty_variadic (ssize_t len, ...)

A simple method for checking multiple managed strings to see if any are empty.

Parameters:

len the number of strings being passed in.

va_list a list of managed strings to check for emptiness

Returns:

true if any of the strings are NULL, uninitialized or empty; false if every string has at least one byte of data.

Definition at line 48 of file data.c.

References `st_data_get()`, and `st_length_get()`.

5.78.4.54 void st_free (stringer_t * s)

Free a managed string.

See also:

[st_valid_free\(\)](#), [st_valid_opts\(\)](#)

Note:

Any managed string to be freed should conform with the validity checks enforced by [st_valid_free\(\)](#)/[st_valid_opts\(\)](#) *NO* managed string should be passed to [st_free\(\)](#) if it was allocated on the stack, or contains foreign data. The following logic is carried out depending on the allocation options of the string to be freed: Jointed placer: Free header, if secure or on heap Free underlying data if it is not foreign Jointed nuller: Free header and underlying data Jointed block: Free header and underlying data Jointed managed: Free header and underlying data Jointed mapped: Free the header and (munmap) underlying data Contiguous nuller: Free header (this includes the underlying data because they are already merged) Contiguous block: Free header (this includes the underlying data because they are already merged) Contiguous managed: Free header (this includes the underlying data because they are already merged)

Parameters:

s the managed string to be freed.

Returns:

This function returns no value.

HIGH: Finish this logic out. Stack allocations are skipped, unless they are jointed. In that case the data is freed unless it carries a foreigner flag. Note its possible for a jointed string to have a NULL data reference. We should be picking up on that too. Contiguous heap buffers are just destroyed.

Do we need to differentiate between stack structures, and heap data blocks needing to be freed, or not?

Definition at line 29 of file allocation.c.

References `block_t`, `BLOCK_T`, `CONTIGUOUS`, `data`, `FOREIGNDATA`, `HEAP`, `JOINTED`, `log_pedantic`, `managed_t`, `MANAGED_T`, `mapped_t`, `MAPPED_T`, `MEMORYBUF`, `mm_free()`, `mm_sec_free()`, `nuller_t`, `NULLER_T`, `placer_t`, `PLACER_T`, `SECURE`, `st_info_opts()`, and `st_valid_free()`.

Referenced by `ar_free()`, `auth_data_fetch()`, `auth_legacy()`, `auth_response()`, `base64_decode_opts()`, `base64_encode_opts()`, `cache_free()`, `cache_set_value()`, `client_print()`, `compress_bzip()`, `compress_lzo()`, `compress_zlib()`, `config_free()`, `config_load_cmdline_settings()`, `config_load_file_settings()`, `config_value_set()`, `contact_business()`, `contact_folder_rename()`, `contact_free()`, `contact_print_form()`, `contact_print_message()`, `decompress_block_lzo()`, `decompress_bzip()`, `decompress_lzo()`, `decompress_zlib()`, `deprecated_hmac_digest()`, `deprecated_scramble_decrypt()`, `deprecated_scramble_encrypt()`, `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `deprecated_symmetric_vector()`, `dkim_start()`, `dspam_check()`, `dspam_train()`, `encrypted_chunk_free()`, `encrypted_chunk_get()`, `encrypted_message_`

free(), ephemeral_chunk_free(), file_temp_handle(), folder_count(), hash_digest(), hex_decode_opts(), hex_encode_opts(), hmac_digest(), http_content_load_directory(), http_content_load_fonts(), http_content_refresh(), http_content_start(), http_data_header_parse_line(), http_data_value_parse(), http_load_file(), http_print_301(), imap_build_array(), imap_command_parser(), imap_fetch_body(), imap_fetch_body_tag(), imap_fetch_bodystructure(), imap_fetch_response_add(), imap_folder_create(), imap_folder_name_escaped(), imap_folder_rename(), imap_id(), imap_list(), imap_login(), imap_lsub(), imap_narrow_folders(), imap_parse_address(), imap_parse_address_part(), imap_parse_literal(), imap_range_build(), imap_search_messages_body(), imap_search_messages_date(), imap_search_messages_header(), imap_search_messages_text(), imap_status(), ip_presentation(), ip_standard(), ip_subnet(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_add_required_headers(), mail_build_signature(), mail_create_message(), mail_discover_encoding(), mail_discover_insertion_point(), mail_discover_type(), mail_extract_tag(), mail_header_fetch_all(), mail_headers(), mail_insert_chunk_base64(), mail_insert_chunk_text(), mail_load_message(), mail_message_cleanup(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_encode_part(), mail_mime_encoding(), mail_mime_generate_boundary(), mail_mime_get_smtp_envelope(), mail_mime_split(), mail_mime_type(), mail_mime_type_group(), mail_mime_type_parameters(), mail_mime_type_sub(), mail_mod_subject(), mail_modify_part(), meta_crypto_keys_create(), meta_data_fetch_all_tags(), meta_data_fetch_keys(), meta_folders_by_name(), meta_folders_name(), mt_free(), naked_message_get(), naked_message_set(), nvp_alloc(), nvp_parse(), org_encrypted_key_get(), org_encrypted_key_set(), org_signet_free(), org_signet_generate(), part_decrypt(), pop_pass(), portal_endpoint_attachments_progress(), portal_endpoint_folders_add(), portal_endpoint_folders_rename(), portal_endpoint_messages_list(), portal_endpoint_messages_send(), portal_endpoint_search(), portal_parse_json_str_array(), portal_print_login(), portal_smtp_create_data(), prime_field_write(), prime_get(), prime_header_write(), prime_key_encrypt(), prime_pem_unwrap(), prime_pem_wrap(), prime_start(), protocol_secure(), register_data_fetch_blocklist(), register_print_captcha(), register_print_message(), register_print_step1(), register_print_step2(), register_print_step3(), register_session_cache(), register_session_get(), relay_free(), relay_set_value(), sanity_check(), scramble_decrypt(), scramble_encrypt(), serial_get(), serial_increment(), serial_reset(), servers_free(), servers_set_value(), sess_get(), signature_tree_add(), signature_tree_alloc(), signature_tree_get(), signature_tree_verify(), slots_key(), smtp_accept_message(), smtp_auth_login(), smtp_auth_plain(), smtp_bounce(), smtp_check_receive_quota(), smtp_data(), smtp_data_outbound(), smtp_data_read(), smtp_fetch_authorization(), smtp_forward_message(), smtp_mail_from(), smtp_rcpt_to(), smtp_reply(), smtp_rollout(), smtp_update_receive_stats(), spool_cleanup(), spool_mktemp(), spool_start(), spool_stop(), st_append_out(), st_bitwise(), st_cleanup_variadic(), st_not(), st_replace(), st_vaprint_opts(), stacie_decrypt(), stacie_encrypt(), stamp_counter_check(), statistics_process(), symmetric_decrypt(), symmetric_encrypt(), symmetric_vector(), teacher_add_cookie(), teacher_data_get(), teacher_data_save(), teacher_print_form(), teacher_print_message(), user_encrypted_key_get(), user_encrypted_key_set(), user_signet_free(), virus_stop(), warehouse_fetch_patterns(), and zbase32_decode().

5.78.4.55 stringer_t* st_import (const void *s, size_t len)

Create a new (contiguous managed) managed string on the heap to hold a copy of the specified data buffer.

See also:

[st_import_opts\(\)](#)

Parameters:

- s* the address of the buffer to be duplicated.
- len* the length, in bytes, of the copied buffer.

Returns:

NULL on failure, or a pointer to the newly allocated managed string on success.

Definition at line 267 of file allocation.c.

References CONTIGUOUS, HEAP, MANAGED_T, and st_import_opts().

Referenced by cache_get(), con_reverse_lookup(), deprecated_ecies_key_public_hex(), dspam_check(), ecies_key_public_hex(), file_temp_handle(), http_data_header_parse_line(), http_load_file(), imap_fetch_body(), imap_fetch_body_tag(), imap_fetch_bodystructure(), imap_fetch_message(), imap_fetch_return_header(), imap_parse_astring(), imap_parse_nstring(), imap_search_messages_date(), mail_add_outbound_headers(), mail_add_required_headers(), mail_header_fetch_all(), mail_load_header(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_type_group(), mail_mime_type_parameters_key(), mail_mime_type_parameters_value(), mail_mime_type_sub(), meta_folders_name(), nvp_parse(), org_signet_set(), pop_user_parse(), portal_endpoint_attachments_add(), portal_endpoint_folders_add(), portal_endpoint_folders_rename(), portal_endpoint_messages_list(), portal_parse_json_str_array(), portal_upload(), register_captcha_generate(), res_field_string(), servers_network_start(), smtp_parse_auth(), smtp_parse_helo_domain(), smtp_parse_mail_from_path(), smtp_parse_rcpt_to(), tank_load(), teacher_add_cookie(), user_request_set(), user_signet_set(), xml_dump_doc(), xml_get_xpath_st(), and xml_node_get_content_st().

5.78.4.56 stringer_t* st_import_opts (uint32_t *opts*, const void * *s*, size_t *len*)

Create a new (contiguous managed) managed string on the heap to hold a copy of the specified data buffer.

Parameters:

opts the options value of the resulting managed string that will be allocated for the caller.

s the address of the buffer to be duplicated.

len the length, in bytes, of the copied buffer.

Returns:

NULL on failure, or a pointer to the newly allocated managed string on success.

Definition at line 236 of file allocation.c.

References log_pedantic, MEMORYBUF, mm_copy(), st_alloc_opts(), st_data_get(), st_info_opts(), st_length_set(), and st_valid_destination().

Referenced by deprecated_ecies_key_private_hex(), ecies_key_private_hex(), and st_import().

5.78.4.57 const chr_t* st_info_allocator (uint32_t *opts*)

Get a readable description of a managed string's allocator type.

Parameters:

opts a value containing the managed string's option mask.

Returns:

a pointer to a null-terminated string containing a description of the managed string's allocator type.

Definition at line 96 of file info.c.

References HEAP, SECURE, st_option_allocators, and STACK.

Referenced by st_info_opts().

5.78.4.58 const chr_t* st_info_layout (uint32_t *opts*)

Get a readable description of a managed string's data layout.

Parameters:

opts a value containing the managed string's option mask.

Returns:

a pointer to a null-terminated string containing a description of the managed string's data layout.

Definition at line 75 of file info.c.

References CONTIGUOUS, JOINTED, and st_option_layouts.

Referenced by st_info_opts().

5.78.4.59 `chr_t* st_info_opts (uint32_t opts, chr_t * s, size_t len)`

Get a readable description of all of a managed string's option flags.

Parameters:

- opts* a value containing the managed string's option mask.
- s* a pointer to a null-terminated string to receive the options description.
- len* the length, in bytes, of the description output buffer.

Returns:

NULL on failure, or a pointer to the output buffer containing the managed string's options description on success.

LOW: Turn this into a loop.

Definition at line 122 of file info.c.

References FOREIGNDATA, NULLER, st_append, st_char_get(), st_cleanup, st_empty, st_info_allocator(), st_info_layout(), st_info_type(), st_length_int(), and st_option_flags.

Referenced by st_alloc_opts(), st_avail_get(), st_avail_set(), st_data_get(), st_data_set(), st_dupe_opts(), st_free(), st_import_opts(), st_length_get(), st_length_set(), st_merge_opts(), st_opt_set(), st_opt_test(), st_realloc(), st_set(), st_vaprint_opts(), st_vsprint(), and st_wipe().

5.78.4.60 `const chr_t* st_info_type (uint32_t opts)`

Get a readable description of a managed string's data type.

Parameters:

- opts* a value containing the managed string's option mask.

Returns:

a pointer to a null-terminated string containing a description of the managed string's data type.

Definition at line 42 of file info.c.

References BLOCK_T, CONSTANT_T, MANAGED_T, MAPPED_T, NULLER_T, PLACER_T, and st_option_types.

Referenced by st_info_opts().

5.78.4.61 `size_t st_length_get (stringer_t * s)`

Return the length of the data in a managed string.

Parameters:

- s* the input managed string.

Returns:

0 on failure, or the length of the managed string's data in bytes.

Definition at line 15 of file length.c.

References block_t, BLOCK_T, constant_t, CONSTANT_T, data, log_pedantic, managed_t, MANAGED_T, mapped_t, MAPPED_T, MEMORYBUF, nuller_t, NULLER_T, placer_t, PLACER_T, st_info_opts(), and st_valid_opts().

Referenced by _serialize_ec_pubkey(), aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), aes_cipher_key(), aes_tag_shard(), aes_vector_shard(), alert_alloc(), alias_alloc(), auth_challenge(), auth_data_fetch(), auth_data_update_legacy(), auth_login(), auth_response(), auth_sanitise_address(), auth_sanitise_username(), base64_decode(), base64_decode_mod(),

base64_encode(), base64_encode_mod(), base64_encode_wrap(), cache_add(), cache_append(), cache_config(), cache_decrement(), cache_delete(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_set_value(), cache_silent_add(), chunk_buffer_read(), chunk_header_read(), client_read(), client_read_line(), compress_bzip(), compress_import(), compress_lzo(), compress_zlib(), con_read(), con_read_line(), con_write_st(), config_value_set(), contact_alloc(), contact_business_valid_email(), contact_detail_alloc(), contact_detail_delete(), contact_detail_upsert(), contact_folder_create(), contact_folder_rename(), contact_insert(), contact_print_form(), contact_print_message(), contact_update(), decompress_block_lzo(), deprecated_hmac_digest(), deprecated_scramble_decrypt(), deprecated_scramble_encrypt(), deprecated_scramble_import(), deprecated_symmetric_key(), deserialize_int16(), deserialize_int32(), deserialize_int64(), deserialize_ns(), deserialize_st(), deserialize_uint16(), deserialize_uint32(), deserialize_uint64(), dkim_signature_create(), dkim_signature_verify(), domain_alloc(), double_conv(), dspam_train(), ed25519_private_set(), ed25519_public_set(), ed25519_sign(), ed25519_verify(), encrypted_chunk_get(), encrypted_chunk_set(), ephemeral_chunk_get(), ephemeral_chunk_set(), float_conv(), get_header_value_noopt(), hash_digest(), hex_decode_opts(), hex_decode_st(), hex_encode_opts(), hex_encode_st(), hex_encode_st_debug(), hmac_digest(), http_body(), http_data_header_parse(), http_data_value_decode(), http_load_file(), http_page_get(), http_parse_context(), http_parse_header(), http_parse_pairs(), http_response(), imap_append_message(), imap_build_array(), imap_command_parser(), imap_copy(), imap_fetch_body(), imap_fetch_body_header(), imap_fetch_bodystructure(), imap_fetch_envelope(), imap_fetch_message(), imap_fetch_parse_partial(), imap_folder_compare(), imap_id(), imap_init(), imap_narrow_folders(), imap_parse_address_breaker(), imap_parse_address_part(), imap_parse_address_put(), imap_process(), imap_search_messages_date(), imap_search_messages_header(), imap_starttls(), imap_status(), imap_valid_folder_name(), int16_conv_st(), int32_conv_st(), int64_conv_st(), int8_conv_st(), ip_subnet(), line_pl_st(), magma_folder_alloc(), magma_folder_insert(), magma_folder_rename(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_add_required_headers(), mail_build_signature(), mail_db_insert_duplicate_message(), mail_db_insert_message(), mail_discover_encoding(), mail_discover_insertion_point(), mail_discover_type(), mail_get_boundary(), mail_get_chunk(), mail_header_end(), mail_header_fetch_all(), mail_header_fetch_cleaned(), mail_header_pop(), mail_headers(), mail_insert_chunk_base64(), mail_insert_chunk_text(), mail_load_message_top(), mail_message(), mail_message_cleanup(), mail_mime_boundary(), mail_mime_child(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_count(), mail_mime_encoding(), mail_mime_header(), mail_mime_part(), mail_mime_type(), mail_mime_type_group(), mail_mime_type_parameters_key(), mail_mime_type_parameters_value(), mail_mime_type_sub(), mail_mime_update(), mail_mod_subject(), mail_modify_part(), mail_signature_add(), mail_store_message_data(), message_alloc(), meta_data_delete_tag(), meta_data_fetch_shard(), meta_data_insert_folder(), meta_data_insert_keys(), meta_data_insert_shard(), meta_data_insert_tag(), meta_data_update_folder_name(), meta_folder_stats_tag_alloc(), meta_update_realms(), mt_get_length(), naked_message_get(), naked_message_set(), nvp_parse(), org_signet_fingerprint(), org_signet_set(), org_signet_verify(), pl_length_get(), pop_num_parse(), pop_pass_parse(), pop_retr(), pop_top(), pop_top_parse(), pop_user_parse(), portal_endpoint_alert_acknowledge(), portal_endpoint_attachments_add(), portal_endpoint_attachments_progress(), portal_endpoint_attachments_remove(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_edit(), portal_endpoint_contacts_list(), portal_endpoint_contacts_load(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), portal_endpoint_folders_add(), portal_endpoint_folders_list(), portal_endpoint_folders_remove(), portal_endpoint_folders_rename(), portal_endpoint_folders_tags(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_load(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), portal_endpoint_search(), portal_message_attachments(), portal_print_login(), portal_settings_changepass(), portal_upload(), portal_validate_request(), prime_key_decrypt(), prime_pem_unwrap(), prime_pem_wrap(), prime_reader_open(), prime_unpack(), qp_decode(), rand_write(), register_abuse_check_blocklist(), register_business_validate_password(), register_business_validate_username(), register_captcha_generate(), register_data_check_username(), register_data_insert_user(), register_print_captcha(), register_print_message(), register_print_step1(), register_print_step2(), register_print_step3(), relay_config(), relay_set_value(), scramble_decrypt(), scramble_encrypt(), scramble_import(), secp256k1_private_set(), secp256k1_public_set(), serialize_int16(), serialize_int32(), serialize_int64(), serialize_ns(), serialize_st(), serialize_uint16(), serialize_uint32(), serialize_uint64(), servers_config(), servers_set_value(), signature_full_get(), signature_full_verify(), signature_tree_add(), signature_tree_get(), signature_tree_verify(), slots_get(), slots_key(), slots_set(), smtp_accept_message(), smtp_auth_plain(), smtp_check_authorized_from(), smtp_check_filters(), smtp_check_greylist(), smtp_check_receive_quota(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), smtp_client_send_rcptto(), smtp_data_outbound(), smtp_fetch_authorization(), smtp_fetch_inbound(), smtp_forward_message(), smtp_insert_spamsig(), smtp_update_receive_stats(), sql_insert_conn(), sql_num_rows_conn(), sql_query_conn(), sql_query_res_conn(), sql_write_conn(), st_append_opts(), st_append_out(), st_avail_get(), st_bitwise(), st_dupe_opts(), st_empty_out(), st_empty_variadic(), st_length_int(), st_merge_opts(), st_not(), st_populated_variadic(), st_realloc(), st_trim(), stacie_decrypt(), stacie_encrypt(), stacie_realm_cipher(), stacie_realm_tag(), stacie_realm_vector(), statistics_process(), symmetric_key(), tank_store(), teacher_print_form(), teacher_print_message(), teacher_process(), tok_get_count_st(), tok_get_st(), tok_pop_init_st(), uint16_conv_st(), uint16_get_no(), uint24_get_no(), uint32_conv_st(), uint32_get_no(), uint64_conv_st(), uint8_conv_st(), url_decode(), user_config_delete(), user_config_entry_alloc(), user_config_upsert(), user_request_generate(), user_request_rotation(), user_request_set(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_set(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), user_signet_verify_self(), and virus_check().

5.78.4.62 `int_t st_length_int (stringer_t * s)`

Length. Return the length of a managed string as an integer.

Parameters:

s the managed string as type `stringer_t *`.

Returns:

the string length as an integer, capped at `INT_MAX`.

Definition at line 70 of file `length.c`.

References `log_pedantic`, `st_length_get()`, and `st_valid_opts()`.

Referenced by `api_endpoint_auth()`, `auth_data_fetch()`, `auth_login()`, `auth_response()`, `cache_add()`, `cache_append()`, `cache_config()`, `cache_decrement()`, `cache_delete()`, `cache_get()`, `cache_get_u64()`, `cache_increment()`, `cache_output_settings()`, `cache_set()`, `cache_set_u64()`, `cipher_name()`, `client_read()`, `client_read_line()`, `client_write()`, `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_file_settings()`, `config_output_value()`, `config_output_value_generic()`, `contact_abuse_checks()`, `contact_abuse_increment_history()`, `contact_detail_delete()`, `contact_detail_upsert()`, `digest_name()`, `dkim_start()`, `dmtip_ehlo()`, `dmtip_helo()`, `dmtip_init()`, `double_conv()`, `float_conv()`, `http_body()`, `http_data_value_parse()`, `http_load_file()`, `http_parse_header()`, `http_parse_method()`, `http_print_301()`, `http_response_allow_cross()`, `http_response_cookie()`, `http_response_header()`, `http_response_options()`, `imap_append()`, `imap_append_message()`, `imap_capability()`, `imap_check()`, `imap_close()`, `imap_command_log_safe()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_id()`, `imap_idle()`, `imap_init()`, `imap_invalid()`, `imap_list()`, `imap_login()`, `imap_logout()`, `imap_lsub()`, `imap_narrow_messages()`, `imap_noop()`, `imap_process()`, `imap_rename()`, `imap_search()`, `imap_select()`, `imap_status()`, `imap_store()`, `imap_subscribe()`, `imap_unsubscribe()`, `lock_get()`, `lock_release()`, `mail_create_directory()`, `mail_load_message()`, `mail_message_path()`, `mail_signature_add()`, `mail_store_message()`, `meta_crypto_keys_create()`, `meta_data_fetch_keys()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_shard()`, `meta_data_insert_keys()`, `meta_data_insert_shard()`, `meta_update_keys()`, `meta_update_realms()`, `pl_length_int()`, `pop_pass()`, `portal_config_entry()`, `portal_endpoint_auth()`, `prime_start()`, `process_kill()`, `protocol_secure()`, `register_business_step1()`, `register_data_insert_user()`, `register_session_cache()`, `register_session_get()`, `relay_config()`, `relay_output_settings()`, `serial_get()`, `serial_increment()`, `serial_reset()`, `servers_config()`, `servers_output_settings()`, `servers_set_value()`, `sess_get()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_bounce()`, `smtp_check_filters()`, `smtp_check_greylist()`, `smtp_check_rbl()`, `smtp_client_connect()`, `smtp_client_send_data()`, `smtp_client_send_mailfrom()`, `smtp_client_send_nullfrom()`, `smtp_client_send_rcptto()`, `smtp_ehlo()`, `smtp_fetch_authorization()`, `smtp_fetch_inbound()`, `smtp_helo()`, `smtp_init()`, `smtp_rcpt_to()`, `smtp_reply()`, `spf_check()`, `spool_check()`, `spool_cleanup()`, `spool_mktemp()`, `spool_start()`, `spool_stop()`, `sql_query_conn()`, `st_info_opts()`, `stacie_derive_key()`, `stacie_derive_token()`, `user_config_delete()`, `user_config_upsert()`, and `virus_start()`.

5.78.4.63 `size_t st_length_set (stringer_t * s, size_t len)`

Set the data length for a managed string that supports data length tracking.

Parameters:

s the input managed string.

len the new value of the managed string's data length.

Returns:

0 on error, or the new data length on success.

Definition at line 99 of file `length.c`.

References `log_pedantic`, `managed_t`, `MANAGED_T`, `mapped_t`, `MAPPED_T`, `MEMORYBUF`, `placer_t`, `PLACER_T`, `st_info_opts()`, and `st_valid_tracked()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `auth_sanitize_address()`, `auth_sanitize_username()`, `base64_decode()`, `base64_decode_mod()`, `base64_encode()`, `base64_encode_mod()`, `base64_encode_wrap()`, `chunk_header_write()`, `client_read()`, `client_read_line()`, `con_read()`, `con_read_line()`, `contact_name()`, `decompress_block_lzo()`, `decompress_bzip()`, `decompress_lzo()`, `decompress_zlib()`, `deprecated_hmac_digest()`, `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `deprecated_symmetric_key()`, `deprecated_symmetric_vector()`, `deserialize_st()`, `ed25519_private_get()`, `ed25519_public_get()`,

ed25519_sign(), encrypted_chunk_set(), ephemeral_chunk_set(), file_load(), file_read(), hash_digest(), hex_decode_st(), hex_encode_st(), hex_encode_st_debug(), hmac_digest(), http_data_value_decode(), http_load_file(), http_parse_header(), imap_build_array(), imap_parse_address_part(), imap_parse_address_put(), imap_parse_literal(), imap_parse_qstring(), imap_starttls(), imap_valid_folder_name(), ip_presentation(), ip_reversed(), ip_standard(), mail_add_required_headers(), mail_extract_address(), mail_extract_tag(), mail_header_fetch_cleaned(), mail_load_message(), mail_load_message_top(), mail_message_cleanup(), mail_mime_generate_boundary(), pop_starttls(), prime_field_write(), prime_header_write(), qp_decode(), rand_choices(), rand_write(), secp256k1_compute_kek(), secp256k1_private_get(), secp256k1_public_get(), serialize_int16(), serialize_int32(), serialize_int64(), serialize_ns(), serialize_st(), serialize_uint16(), serialize_uint32(), serialize_uint64(), smtp_data_finish(), smtp_data_read(), smtp_starttls(), st_append_opts(), st_append_out(), st_bitwise(), st_copy_in(), st_dupe_opts(), st_import_opts(), st_merge_opts(), st_not(), st_nullify(), st_replace(), st_set(), st_trim(), st_vaprint_opts(), st_vsprint(), st_wipe(), st_write_variadic(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_encrypt(), symmetric_decrypt(), symmetric_encrypt(), symmetric_key(), symmetric_vector(), time_print_gmt(), time_print_local(), uint16_put_no(), uint24_put_no(), uint32_put_no(), url_decode(), url_encode(), zbase32_decode(), and zbase32_encode().

5.78.4.64 stringer_t* st_merge_opts (uint32_t *opts*, chr_t **format*, ...)

Concatenate a variable number of user-supplied multi-type strings.

Parameters:

opts the options value of the resulting managed string that will be allocated for the caller.

format a format string consisting of the characters 's' for specifying that the next variable argument will be a managed string or 'n' if it is a null-terminated string.

... a variable argument list containing the strings to be concatenated to one another.

Returns:

a newly allocated string containing the concatenations of all specified managed and null-terminated strings.

Definition at line 120 of file allocation.c.

References length, log_pedantic, MEMORYBUF, mm_copy(), ns_empty(), ns_length_get(), st_alloc_opts(), st_char_get(), st_cleanup, st_data_get(), st_empty, st_info_opts(), st_length_get(), st_length_set(), st_valid_destination(), and st_valid_tracked().

Referenced by mail_add_forward_headers(), mail_add_inbound_headers(), mail_add_outbound_headers(), mail_add_required_headers(), and spool_path().

5.78.4.65 stringer_t* st_nullify (chr_t **input*, size_t *len*)

Create a null-terminated string out of a block of memory and return it as a newly allocated nuller.

Parameters:

input a pointer to the data block to be copied.

len the size, in bytes, of the data block to be copied before being terminated with a null byte.

Returns:

NULL on failure, or a pointer to the new null-terminated string on success.

Definition at line 841 of file allocation.c.

References CONTIGUOUS, HEAP, MANAGED_T, mm_copy(), st_alloc_opts(), st_data_get(), and st_length_set().

Referenced by portal_message_body().

5.78.4.66 `uint32_t st_opt_get (stringer_t * s)`

[opts.c](#) [opts.c](#)

Parameters:

s the input string.

Returns:

the options in use by the provided string.

Definition at line 15 of file `opts.c`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `chunk_header_write()`, `ed25519_private_get()`, `ed25519_public_get()`, `ed25519_sign()`, `meta_data_fetch_shard()`, `org_key_get()`, `org_signet_get()`, `prime_field_write()`, `prime_header_write()`, `rand_choices()`, `secp256k1_compute_kek()`, `secp256k1_private_get()`, `secp256k1_public_get()`, `st_write_variadic()`, `stacie_create_nonce()`, `stacie_create_salt()`, `stacie_create_shard()`, `user_key_get()`, `user_request_get()`, and `user_signet_get()`.

5.78.4.67 `int_t st_opt_set (stringer_t * s, uint32_t opt, bool_t enabled)`

Enable or disable a set of option(s) for a managed string, with validity testing.

Parameters:

s the input managed string.

opt the bitmask of option(s) to be enabled or disabled for the managed string.

enabled if true, set the option(s); if false, disable them.

Returns:

-1 on error, 0 if successfully disabled, or 1 if successfully enabled.

Definition at line 33 of file `opts.c`.

References `log_pedantic`, `MEMORYBUF`, `st_info_opts()`, and `st_valid_opts()`.

Referenced by `contact_name()`.

5.78.4.68 `bool_t st_opt_test (stringer_t * s, uint32_t opt)`

Check to see if the managed string has the specified option enabled.

Parameters:

s the managed string to be checked. *opt* a bitmask of the managed string options to be tested.

Returns:

-1 on error, 0 if *opt* is not set, and 1 if *opt* is set.

Definition at line 64 of file `opts.c`.

References `log_pedantic`, `MEMORYBUF`, `st_info_opts()`, and `st_valid_opts()`.

Referenced by `contact_free()`, and `contact_name()`.

5.78.4.69 stringer_t* st_output (stringer_t * *output*, size_t *len*)

Return a suitable managed string to store data of a specified length.

Note:

If a NULL output string is specified, one will be allocated for the caller.

Parameters:

output the input managed string (can be NULL).

len the size, in bytes, of the requested data buffer.

Returns:

NULL on failure or if input was not large enough or a pointer to a validated output managed string.

Definition at line 812 of file allocation.c.

References log_pedantic, st_alloc(), st_avail_get(), and st_valid_destination().

Referenced by chunk_header_write(), deprecated_symmetric_key(), deprecated_symmetric_vector(), encrypted_chunk_get(), prime_field_write(), prime_header_write(), prime_pem_wrap(), slots_key(), symmetric_key(), symmetric_vector(), tls_cipher(), and tls_error().

5.78.4.70 bool_t st_populated_variadic (ssize_t *len*, ...)

A simple method for checking multiple managed strings to ensure all of the provided strings have contain at least one character of data.

Parameters:

len the number of strings being passed in.

va_list a list of managed strings to be validated.

Returns:

true if all of the strings are populated with at least one byte of data, otherwise false.

Definition at line 19 of file data.c.

References st_data_get(), and st_length_get().

5.78.4.71 stringer_t stringer_t stringer_t* st_quick (stringer_t * *s*, chr_t * *format*, ...)**5.78.4.72 stringer_t* st_realloc (stringer_t * *s*, size_t *len*)**

Reallocate a managed string to a specified size.

Note:

The caller can request a new size that is either smaller (truncated) or larger than the original string. Only these types of strings can be reallocated: nuller, block, managed, and mapped. The following logic is applied to string reallocation requests: For jointed strings, a new data buffer is allocated of the requested length, the original contents copied in, the data field of the new string is set, and the original data buffer freed. The stringer header returned to the caller resides at the same address as the original. For contiguous strings, a new data buffer is allocated for both the requested length and the new stringer header, and the data of both parts are copied from the original string to the resulting one. For block strings, the length field of the resulting string is set to the requested length. For managed strings, the length field of the resulting string is set to the original length, but the available field is adjusted accordingly. For mapped strings, an aligned mremap() call is made, and both the length and available fields are set to the new length.

Parameters:

s the managed string to have its size reallocated.

len the new size, in bytes, of the resulting managed string.

Returns:

NULL on failure or a pointer to the newly reallocated managed string on success.

Definition at line 656 of file allocation.c.

References align(), block_t, BLOCK_T, CONTIGUOUS, data, JOINTED, length, log_pedantic, magma, managed_t, MANAGED_T, mapped_t, MAPPED_T, magma_t::memory, MEMORYBUF, mm_alloc(), mm_copy(), mm_free(), mm_sec_alloc(), mm_sec_free(), nuller_t, NULLER_T, magma_t::page_length, magma_t::secure, SECURE, st_info_opts(), st_length_get(), st_valid_opts(), and system_ulimit_cur().

Referenced by serialize_int16(), serialize_int32(), serialize_int64(), serialize_ns(), serialize_st(), serialize_uint16(), serialize_uint32(), serialize_uint64(), smtp_data_read(), st_append_opts(), and st_append_out().

5.78.4.73 int_t st_replace (stringer_t **target, stringer_t *pattern, stringer_t *replacement)

[replace.c](#)

Definition at line 19 of file replace.c.

References log_pedantic, PLACER, st_alloc(), st_char_get(), st_cmp_cs_starts(), st_data_get(), st_empty_out(), st_free(), st_length_set(), and st_valid_free().

Referenced by contact_business(), imap_folder_create(), imap_folder_name_escaped(), imap_folder_rename(), pop_retr(), pop_top(), register_print_message(), register_print_step1(), register_print_step2(), smtp_client_send_data(), smtp_forward_message(), smtp_relay_message(), and teacher_process().

5.78.4.74 stringer_t* st_set (stringer_t *s, uint8_t set, size_t len)

Sets a block of memory to a specified value.

Parameters:

s the managed string to be overwritten with the value of set.

set the byte value to be write into the string.

len the number of bytes to set to the provided value.

Returns:

a pointer to the block of memory passed to the function.

Definition at line 246 of file data.c.

References log_pedantic, MEMORYBUF, mm_set(), st_avail_get(), st_data_get(), st_info_opts(), st_length_set(), st_valid_opts(), and st_valid_tracked().

Referenced by encrypted_chunk_get(), encrypted_chunk_set(), and slots_key().

5.78.4.75 stringer_t stringer_t stringer_t size_t st_sprint (stringer_t *s, chr_t *format, ...)

5.78.4.76 stringer_t* st_swap (stringer_t *target, uchr_t pattern, uchr_t replacement)

Replace all instances of one character in a managed string with another.

Parameters:

target the input string which will be transformed by the character replacement.

pattern the character to be searched and replaced in the target string.

replacement the character to be substituted for the pattern character in the target string.

Returns:

a pointer to the target managed string.

Definition at line 109 of file replace.c.

References `log_check`, `log_pedantic`, and `st_empty_out()`.

Referenced by `process_kill()`.

5.78.4.77 `uchr_t* st_uchar_get (stringer_t * s)`

Retrieve an unsigned character pointer to a managed string's data buffer.

See also:

[st_data_get\(\)](#)

Parameters:

s the input managed string.

Returns:

NULL on failure or for an improperly constructed string; otherwise, a pointer to the string's data.

Definition at line 206 of file data.c.

References `st_data_get()`.

Referenced by `dkim_signature_create()`, `http_parse_origin()`, and `prime_pem_unwrap()`.

5.78.4.78 `bool_t st_used_variadic (ssize_t len, ...)`**5.78.4.79 `bool_t st_valid_append (uint32_t opts)`**

Determine whether the managed string options allow for data appending.

Parameters:

opts the options value for the managed string.

Returns:

true if permitted, or false if options are invalid, the string is not jointed, or either managed or mapped.

Definition at line 52 of file validate.c.

References `JOINTED`, `MANAGED_T`, `MAPPED_T`, and `st_valid_opts()`.

Referenced by `st_append_opts()`.

5.78.4.80 `bool_t st_valid_avail (uint32_t opts)`

Determine whether a managed string tracks the total allocated buffer size.

Parameters:

opts the managed string's options value.

Returns:

false if buffer size isn't tracked; true otherwise (the managed string is managed or mapped).

Definition at line 123 of file validate.c.

References MANAGED_T, MAPPED_T, and st_valid_opts().

Referenced by base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), deprecated_hmac_digest(), hash_digest(), hex_decode_st(), hex_encode_st(), hmac_digest(), ip_subnet(), rand_write(), st_avail_set(), st_bitwise(), and st_not().

5.78.4.81 bool_t st_valid_destination (uint32_t opts)

Determine whether the managed string options allow for a data store operation.

Parameters:

opts the options value for the managed string.

Returns:

true if permitted, or false if options are invalid, or indicate the managed string is constant.

Definition at line 35 of file validate.c.

References CONSTANT_T, and st_valid_opts().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), deprecated_hmac_digest(), ed25519_private_get(), ed25519_public_get(), ed25519_sign(), hash_digest(), hex_decode_st(), hex_encode_st(), hmac_digest(), ip_presentation(), ip_reversed(), ip_standard(), ip_subnet(), meta_data_fetch_shard(), org_key_get(), org_signet_get(), rand_choices(), rand_write(), secp256k1_compute_kek(), secp256k1_private_get(), secp256k1_public_get(), st_append_out(), st_bitwise(), st_import_opts(), st_merge_opts(), st_not(), st_output(), st_vaprint_opts(), st_vsprint(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), user_key_get(), user_request_get(), and user_signet_get().

5.78.4.82 bool_t st_valid_free (uint32_t opts)

Determine whether a managed string is allowed to be freed.

Parameters:

opts the options value of the managed string to be evaluated.

Returns:

true if the managed string can be freed; false otherwise (it is allocated on the stack or contains foreign data).

Definition at line 72 of file validate.c.

References FOREIGNDATA, st_valid_opts(), and STACK.

Referenced by st_free(), and st_replace().

5.78.4.83 bool_t st_valid_jointed (uint32_t opts)

Determine whether the managed string options are a validly jointed.

Parameters:

opts the options value to test.

Returns:

false if the options are invalid or if the string isn't jointed; true if the string is jointed.

Definition at line 106 of file validate.c.

References JOINTED, and st_valid_opts().

Referenced by st_data_set().

5.78.4.84 bool_t st_valid_opts (uint32_t *opts*)

Check to see that a managed string has a valid combination of allocation options.

Note:

The following rules are enforced: 1. Each managed string must only be one of the following: a. constant, nuller, block, placer, managed, or mapped. b. jointed or contiguous. c. allocated on the stack, heap, or secure. 2. A placer cannot be contiguous. 3. A constant must be contiguous and be allocated on the stack. 4. Mapped strings must be contiguous and on the heap.

Parameters:

opts the managed string option mask to be validated.

Returns:

true if the options represent valid managed string allocation options, or false if they do not.

Definition at line 149 of file validate.c.

References bitwise_count(), BLOCK_T, CONSTANT_T, CONTIGUOUS, HEAP, JOINTED, MANAGED_T, MAPPED_T, NULLER_T, PLACER_T, SECURE, and STACK.

Referenced by st_alloc_opts(), st_avail_get(), st_data_get(), st_dupe_opts(), st_length_get(), st_length_int(), st_opt_set(), st_opt_test(), st_realloc(), st_set(), st_valid_append(), st_valid_avail(), st_valid_destination(), st_valid_free(), st_valid_jointed(), st_valid_placer(), st_valid_tracked(), and st_wipe().

5.78.4.85 bool_t st_valid_placer (uint32_t *opts*)

A sanity check to determine whether the managed string is a valid placer.

Note:

The following criteria must be satisfied: placer bit set, jointed bit set, either stack, heap, or secure set, and no other bits other than these mentioned should be set.

Parameters:

opts the managed string options value to be evaluated.

Returns:

true if the option value reflects a valid placer, or false otherwise.

Definition at line 17 of file validate.c.

References HEAP, JOINTED, PLACER_T, SECURE, st_valid_opts(), and STACK.

5.78.4.86 bool_t st_valid_tracked (uint32_t *opts*)

Determine whether a managed string provides for data length tracking.

Parameters:

opts the options value of the managed string.

Returns:

false if no length tracking is available; true if there is (string is a placer, managed, or mapped).

Definition at line 89 of file validate.c.

References MANAGED_T, MAPPED_T, PLACER_T, and st_valid_opts().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), base64_decode(), base64_decode_mod(), base64_encode(), base64_encode_mod(), base64_encode_wrap(), chunk_header_write(), deprecated_hmac_digest(), deprecated_symmetric_key(), deprecated_symmetric_vector(), ed25519_private_get(), ed25519_public_get(), ed25519_sign(), hash_digest(), hex_decode_st(), hex_encode_st(), hmac_digest(), ip_presentation(), ip_reversed(), ip_standard(), prime_field_write(), prime_header_write(), rand_choices(), rand_write(), secp256k1_compute_kek(), secp256k1_private_get(), secp256k1_public_get(), st_dupe_opts(), st_length_set(), st_merge_opts(), st_set(), st_vaprint_opts(), st_vsprint(), st_wipe(), st_write_variadic(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), symmetric_key(), and symmetric_vector().

5.78.4.87 stringer_t stringer_t stringer_t size_t stringer_t* st_vaprint_opts (uint32_t *opts*, chr_t **format*, va_list *args*)

Return a managed string containing sprintf()-style formatted data.

Parameters:

opts an options value to be passed to the allocation of the resulting managed string.

format a format string for the output string data.

args a variable argument list of parameters to be formatted as output.

Returns:

NULL on failure or a managed string containing the final formatted data on success.

Definition at line 100 of file print.c.

References length, log_pedantic, MEMORYBUF, st_alloc_opts(), st_data_get(), st_free(), st_info_opts(), st_length_set(), st_valid_destination(), and st_valid_tracked().

Referenced by st_aprint(), and st_aprint_opts().

5.78.4.88 size_t st_vsprint (stringer_t **s*, chr_t **format*, va_list *args*)

Print to a managed string and return the number of bytes written.

See also:

vsnprintf()

Note:

If the destination string pointer is NULL the function will simply return how much room would have been necessary.

Parameters:

s a pointer to the managed string that will receive the output of the print operation.

format a format string specifying the arguments of the print operation.

args a va_list containing the parameters to the print format string.

Returns:

-1 on failure, or the number of characters printed to the string, excluding the terminating null byte.

Definition at line 19 of file print.c.

References length, log_pedantic, MEMORYBUF, st_avail_get(), st_data_get(), st_info_opts(), st_length_set(), st_valid_destination(), and st_valid_tracked().

Referenced by st_quick(), and st_sprint().

5.78.4.89 void st_wipe (stringer_t * s)

Wipe all of a managed string's allocated memory and if applicable, reset its length.

Parameters:

s the managed string to be wiped.

Returns:

This function returns no value.

Definition at line 216 of file data.c.

References log_pedantic, MEMORYBUF, mm_wipe(), st_avail_get(), st_data_get(), st_info_opts(), st_length_set(), st_valid_opts(), and st_valid_tracked().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), ed25519_private_get(), ed25519_public_get(), ip_standard(), org_key_get(), org_signet_get(), secp256k1_private_get(), user_key_get(), user_request_get(), and user_signet_get().

5.78.4.90 int_t st_write_variadic (stringer_t * output, ssize_t count, ...)

Write a variable number of user-supplied strings into a buffer.

Parameters:

... a variable argument list containing the strings to be written one after the other.

Returns:

the number of bytes written into the buffer.

Definition at line 182 of file print.c.

References data, mm_copy(), st_avail_get(), st_data_get(), st_empty_out(), st_length_set(), st_opt_get(), and st_valid_tracked().

5.78.5 Variable Documentation

5.78.5.1 block_t

Definition at line 53 of file strings.h.

Referenced by st_alloc_opts(), st_data_get(), st_data_set(), st_free(), st_length_get(), and st_realloc().

5.78.5.2 constant_t

Definition at line 42 of file strings.h.

Referenced by st_data_get(), and st_length_get().

5.78.5.3 managed_t

Definition at line 66 of file strings.h.

Referenced by st_alloc_opts(), st_avail_get(), st_avail_set(), st_data_get(), st_data_set(), st_free(), st_length_get(), st_length_set(), and st_realloc().

5.78.5.4 mapped_t

Definition at line 74 of file strings.h.

Referenced by st_alloc_opts(), st_avail_get(), st_avail_set(), st_data_get(), st_free(), st_length_get(), st_length_set(), and st_realloc().

5.78.5.5 nuller_t

Definition at line 47 of file strings.h.

Referenced by st_alloc_opts(), st_data_get(), st_data_set(), st_free(), st_length_get(), and st_realloc().

5.78.5.6 placer_t

Definition at line 59 of file strings.h.

Referenced by __attribute__(), aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), aes_cipher_key(), aes_tag_shard(), aes_vector_shard(), alert_alloc(), alias_alloc(), api_endpoint_register(), ar_field_pl(), build_version_major(), build_version_minor(), build_version_patch(), cache_config(), contact_alloc(), contact_detail_alloc(), domain_alloc(), ephemeral_chunk_set(), get_header_opt(), get_header_value_noopt(), http_data_value_parse(), http_parse_context(), http_parse_header(), http_parse_method(), http_parse_pairs(), http_response_allow_cross(), imap_build_array(), imap_fetch_body(), imap_fetch_body_header(), imap_fetch_body_part(), imap_fetch_envelope(), imap_folder_create(), imap_folder_remove(), imap_folder_rename(), imap_narrow_messages(), imap_parse_address(), imap_parse_address_breaker(), imap_parse_address_part(), imap_search_messages_date(), imap_search_messages_date_compare(), imap_search_messages_header(), imap_search_messages_range(), ip_subnet_st(), lib_load_openssl(), magma_folder_alloc(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_mime_split(), mail_mime_type_parameters(), mail_mime_update(), mail_mod_subject(), mail_modify_part(), message_alloc(), meta_folder_stats_tag_alloc(), multipart_get_boundary(), naked_message_get(), naked_message_set(), nvp_parse(), part_decrypt(), pl_init(), pl_null(), pl_set(), portal_get_upload_attachment(), portal_upload(), prime_pem_unwrap(), prime_unpack_fields(), prime_unpack_validate(), relay_config(), sanity_check(), servers_config(), signature_full_get(), signature_full_verify(), signature_tree_get(), signature_tree_verify(), slots_buffer(), smtp_auth_plain(), smtp_check_authorized_from(), smtp_check_filters(), smtp_client_send_helo(), smtp_fetch_inbound(), smtp_parse_mail_from_path(), smtp_rcpt_to(), spf_check(), st_alloc_opts(), st_data_get(), st_data_set(), st_free(), st_length_get(), st_length_set(), stamp_counter_check(), str_tok_get_bl(), str_tok_get_count_bl(), and user_config_entry_alloc().

5.79 src/core/strings/validate.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t st_valid_placer](#) (uint32_t opts)
A sanity check to determine whether the managed string is a valid placer.
- [bool_t st_valid_destination](#) (uint32_t opts)
Determine whether the managed string options allow for a data store operation.
- [bool_t st_valid_append](#) (uint32_t opts)
Determine whether the managed string options allow for data appending.
- [bool_t st_valid_free](#) (uint32_t opts)
Determine whether a managed string is allowed to be freed.
- [bool_t st_valid_tracked](#) (uint32_t opts)
Determine whether a managed string provides for data length tracking.
- [bool_t st_valid_jointed](#) (uint32_t opts)
Determine whether the managed string options are a validly jointed.
- [bool_t st_valid_avail](#) (uint32_t opts)
Determine whether a managed string tracks the total allocated buffer size.
- [bool_t st_valid_opts](#) (uint32_t opts)
Check to see that a managed string has a valid combination of allocation options.

5.79.1 Function Documentation

5.79.1.1 bool_t st_valid_append (uint32_t opts)

Determine whether the managed string options allow for data appending.

Parameters:

opts the options value for the managed string.

Returns:

true if permitted, or false if options are invalid, the string is not jointed, or either managed or mapped.

Definition at line 52 of file validate.c.

References `JOINTED`, `MANAGED_T`, `MAPPED_T`, and `st_valid_opts()`.

Referenced by `st_append_opts()`.

5.79.1.2 `bool_t st_valid_avail (uint32_t opts)`

Determine whether a managed string tracks the total allocated buffer size.

Parameters:

opts the managed string's options value.

Returns:

false if buffer size isn't tracked; true otherwise (the managed string is managed or mapped).

Definition at line 123 of file validate.c.

References `MANAGED_T`, `MAPPED_T`, and `st_valid_opts()`.

Referenced by `base64_decode()`, `base64_decode_mod()`, `base64_encode()`, `base64_encode_mod()`, `base64_encode_wrap()`, `deprecated_hmac_digest()`, `hash_digest()`, `hex_decode_st()`, `hex_encode_st()`, `hmac_digest()`, `ip_subnet()`, `rand_write()`, `st_avail_set()`, `st_bitwise()`, and `st_not()`.

5.79.1.3 `bool_t st_valid_destination (uint32_t opts)`

Determine whether the managed string options allow for a data store operation.

Parameters:

opts the options value for the managed string.

Returns:

true if permitted, or false if options are invalid, or indicate the managed string is constant.

Definition at line 35 of file validate.c.

References `CONSTANT_T`, and `st_valid_opts()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `base64_decode()`, `base64_decode_mod()`, `base64_encode()`, `base64_encode_mod()`, `base64_encode_wrap()`, `deprecated_hmac_digest()`, `ed25519_private_get()`, `ed25519_public_get()`, `ed25519_sign()`, `hash_digest()`, `hex_decode_st()`, `hex_encode_st()`, `hmac_digest()`, `ip_presentation()`, `ip_reversed()`, `ip_standard()`, `ip_subnet()`, `meta_data_fetch_shard()`, `org_key_get()`, `org_signet_get()`, `rand_choices()`, `rand_write()`, `secp256k1_compute_kek()`, `secp256k1_private_get()`, `secp256k1_public_get()`, `st_append_out()`, `st_bitwise()`, `st_import_opts()`, `st_merge_opts()`, `st_not()`, `st_output()`, `st_vaprint_opts()`, `st_vsprint()`, `stacie_create_nonce()`, `stacie_create_salt()`, `stacie_create_shard()`, `user_key_get()`, `user_request_get()`, and `user_signet_get()`.

5.79.1.4 `bool_t st_valid_free (uint32_t opts)`

Determine whether a managed string is allowed to be freed.

Parameters:

opts the options value of the managed string to be evaluated.

Returns:

true if the managed string can be freed; false otherwise (it is allocated on the stack or contains foreign data).

Definition at line 72 of file validate.c.

References `FOREIGNDATA`, `st_valid_opts()`, and `STACK`.

Referenced by `st_free()`, and `st_replace()`.

5.79.1.5 `bool_t st_valid_joined (uint32_t opts)`

Determine whether the managed string options are a validly jointed.

Parameters:

opts the options value to test.

Returns:

false if the options are invalid or if the string isn't jointed; true if the string is jointed.

Definition at line 106 of file validate.c.

References JOINTED, and st_valid_opts().

Referenced by st_data_set().

5.79.1.6 `bool_t st_valid_opts (uint32_t opts)`

Check to see that a managed string has a valid combination of allocation options.

Note:

The following rules are enforced: 1. Each managed string must only be one of the following: a. constant, nuller, block, placer, managed, or mapped. b. jointed or contiguous. c. allocated on the stack, heap, or secure. 2. A placer cannot be contiguous. 3. A constant must be contiguous and be allocated on the stack. 4. Mapped strings must be contiguous and on the heap.

Parameters:

opts the managed string option mask to be validated.

Returns:

true if the options represent valid managed string allocation options, or false if they do not.

Definition at line 149 of file validate.c.

References bitwise_count(), BLOCK_T, CONSTANT_T, CONTIGUOUS, HEAP, JOINTED, MANAGED_T, MAPPED_T, NULLER_T, PLACER_T, SECURE, and STACK.

Referenced by st_alloc_opts(), st_avail_get(), st_data_get(), st_dupe_opts(), st_length_get(), st_length_int(), st_opt_set(), st_opt_test(), st_realloc(), st_set(), st_valid_append(), st_valid_avail(), st_valid_destination(), st_valid_free(), st_valid_joined(), st_valid_placer(), st_valid_tracked(), and st_wipe().

5.79.1.7 `bool_t st_valid_placer (uint32_t opts)`

A sanity check to determine whether the managed string is a valid placer.

Note:

The following criteria must be satisfied: placer bit set, jointed bit set, either stack, heap, or secure set, and no other bits other than these mentioned should be set.

Parameters:

opts the managed string options value to be evaluated.

Returns:

true if the option value reflects a valid placer, or false otherwise.

Definition at line 17 of file validate.c.

References HEAP, JOINTED, PLACER_T, SECURE, st_valid_opts(), and STACK.

5.79.1.8 `bool_t st_valid_tracked(uint32_t opts)`

Determine whether a managed string provides for data length tracking.

Parameters:

opts the options value of the managed string.

Returns:

false if no length tracking is available; true if there is (string is a placer, managed, or mapped).

Definition at line 89 of file validate.c.

References `MANAGED_T`, `MAPPED_T`, `PLACER_T`, and `st_valid_opts()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `base64_decode()`, `base64_decode_mod()`, `base64_encode()`, `base64_encode_mod()`, `base64_encode_wrap()`, `chunk_header_write()`, `deprecated_hmac_digest()`, `deprecated_symmetric_key()`, `deprecated_symmetric_vector()`, `ed25519_private_get()`, `ed25519_public_get()`, `ed25519_sign()`, `hash_digest()`, `hex_decode_st()`, `hex_encode_st()`, `hmac_digest()`, `ip_presentation()`, `ip_reversed()`, `ip_standard()`, `prime_field_write()`, `prime_header_write()`, `rand_choices()`, `rand_write()`, `secp256k1_compute_kek()`, `secp256k1_private_get()`, `secp256k1_public_get()`, `st_dupe_opts()`, `st_length_set()`, `st_merge_opts()`, `st_set()`, `st_vaprint_opts()`, `st_vsprint()`, `st_wipe()`, `st_write_variadic()`, `stacie_create_nonce()`, `stacie_create_salt()`, `stacie_create_shard()`, `symmetric_key()`, and `symmetric_vector()`.

5.80 src/core/thread/keys.c File Reference

```
#include "magma.h"
```

Functions

- int [tkey_init](#) (pthread_key_t *key, void(*destructor)(void *))
Create a thread-specific data key.
- void * [tkey_get](#) (pthread_key_t key)
Get the calling thread's value for a pthread key.
- int [tkey_set](#) (pthread_key_t key, void *value)
Set the calling thread's value for a pthread key.

5.80.1 Function Documentation

5.80.1.1 void* tkey_get (pthread_key_t key)

Get the calling thread's value for a pthread key. keys.c

See also:

pthread_getspecific()

Parameters:

key the pthread key to be queried.

Returns:

NULL on failure or a pointer to the key's thread-specific data value on success.

Definition at line 35 of file keys.c.

5.80.1.2 int tkey_init (pthread_key_t * key, void(*)(void *) destructor)

Create a thread-specific data key.

See also:

pthread_key_create()

Parameters:

key a pointer to pthread key to be initialized for thread-local storage.

destructor if not NULL, an optional function pointer to be called at thread exit to release the data.

Returns:

0 on success, or an error number on failure.

Definition at line 17 of file keys.c.

References log_pedantic.

5.80.1.3 int tkey_set (pthread_key_t *key*, void * *value*)

Set the calling thread's value for a pthread key.

See also:

pthread_setspecific()

Parameters:

key the pthread key to have its thread-local value modified.

value a pointer to the new thread-specific value of the specified key.

Returns:

0 on success, or an error number on failure.

Definition at line 46 of file keys.c.

References log_pedantic.

5.81 src/providers/dime/signet/keys.c File Reference

```
#include <arpa/inet.h>
#include <sys/socket.h>
#include "core/core.h"
#include "dime/common/misc.h"
#include "dime/signet/keys.h"
#include "dime/signet/signet.h"
#include "providers/cryptography/cryptography.h"
#include "prime/prime.h"
#include "providers/symbols.h"
```

Functions

- int [dime_keys_file_create](#) ([keys_type_t](#) type, [ED25519_KEY](#) *sign_key, [EC_KEY](#) *enc_key, const char *filename)
Creates a keys file with specified signing and encryption keys.
- [ED25519_KEY](#) * [dime_keys_signkey_fetch](#) (char const *filename)
Retrieves the encryption key from the keys file.
- [EC_KEY](#) * [dime_keys_enckey_fetch](#) (char const *filename)
Retrieves the signing key from the keys file.
- int [dime_keys_generate](#) ([keys_type_t](#) type, char **signet_pem, char **key_pem)

5.81.1 Function Documentation

5.81.1.1 [EC_KEY](#) * [dime_keys_enckey_fetch](#) (char const * *filename*)

Retrieves the signing key from the keys file.

Parameters:

filename Null terminated filename string.

Returns:

Pointer to the ed25519 signing key. {free_ed25519_key}

Definition at line 633 of file keys.c.

References [PUBLIC_FUNCTION_IMPLEMENT](#).

5.81.1.2 int [dime_keys_file_create](#) ([keys_type_t](#) type, [ED25519_KEY](#) * *sign_key*, [EC_KEY](#) * *enc_key*, const char * *filename*)

Creates a keys file with specified signing and encryption keys.

Parameters:

type Type of keys file, whether the keys correspond to a user or organizational signet.

sign_key Pointer to the specified ed25519 key, the private portion of which will be stored in the keys file as the signing key.

enc_key Pointer to the specified elliptic curve key, the private portion of which will be stored in the keys file as the encryption key.

filename Pointer to the NULL terminated string containing the filename for the keys file.

Returns:

0 on success, -1 on failure.

Definition at line 586 of file keys.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.81.1.3 int dime_keys_generate (keys_type_t type, char ** *signet_pem*, char ** *key_pem*)

Definition at line 637 of file keys.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.81.1.4 ED25519_KEY* dime_keys_signkey_fetch (char const * *filename*)

Retrieves the encryption key from the keys file.

Parameters:

filename Null terminated filename string.

Returns:

Pointer to the elliptic curve encryption key. {free_ec_key}

Definition at line 620 of file keys.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.82 src/core/thread/mutex.c File Reference

```
#include "magma.h"
```

Functions

- int [mutex_init](#) (pthread_mutex_t *[lock](#), pthread_mutexattr_t *[attr](#))
Initialize a pthread mutex with the given attributes.
- int [mutex_lock](#) (pthread_mutex_t *[lock](#))
Acquire a pthread mutex, blocking if necessary.
- int [mutex_unlock](#) (pthread_mutex_t *[lock](#))
Unlock a pthread mutex.
- int [mutex_destroy](#) (pthread_mutex_t *[lock](#))
Free a pthread mutex.

5.82.1 Function Documentation

5.82.1.1 int mutex_destroy (pthread_mutex_t * *lock*)

Free a pthread mutex. [mutex.c](#)

See also:

[pthread_mutex_destroy\(\)](#)

Parameters:

lock a pointer to the mutex to be freed.

Returns:

0 on success, or an error number on failure.

Definition at line 77 of file [mutex.c](#).

References [log_pedantic](#), and [MEMORYBUF](#).

Referenced by [con_destroy\(\)](#), [meta_free\(\)](#), [pool_free\(\)](#), [protocol_secure\(\)](#), [queue_shutdown\(\)](#), [sess_destroy\(\)](#), [ssl_stop\(\)](#), [stacker_free\(\)](#), and [stats_shutdown\(\)](#).

5.82.1.2 int mutex_init (pthread_mutex_t * *lock*, pthread_mutexattr_t * *attr*)

Initialize a pthread mutex with the given attributes.

See also:

[pthread_mutex_init\(\)](#)

Parameters:

lock a pointer to a mutex that will be initialized.

attr a pointer to a mutex attributes holder, or NULL to use system default values.

Returns:

0 on success, or an error number on failure.

Definition at line 17 of file mutex.c.

References log_pedantic, and MEMORYBUF.

Referenced by con_init(), meta_alloc(), pool_alloc(), queue_init(), ssl_start(), stacker_alloc(), and stats_init().

5.82.1.3 int mutex_lock (pthread_mutex_t * lock)

Acquire a pthread mutex, blocking if necessary.

See also:

pthread_mutex_lock()

Parameters:

lock a pointer to the mutex to be locked.

Returns:

0 on success, or an error number on failure.

Definition at line 37 of file mutex.c.

References log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, and MEMORYBUF.

Referenced by con_decrement_refs(), con_increment_refs(), con_reverse_check(), con_reverse_domain(), con_reverse_enqueue(), con_reverse_status(), dequeue(), dh_exchange_2048(), dh_exchange_4096(), dh_params_generate(), log_disable(), log_enable(), log_internal(), meta_user_ref_add(), meta_user_ref_dec(), meta_user_ref_protocol_total(), meta_user_ref_stamp(), meta_user_ref_total(), mm_sec_alloc(), mm_sec_free(), mm_sec_stats(), pattern_check(), pattern_stop(), pattern_update(), pool_get_failures(), pool_get_obj(), pool_pull(), pool_release(), pool_set_obj(), pool_swap_obj(), portal_endpoint_attachments_add(), portal_endpoint_attachments_remove(), portal_endpoint_messages_compose(), portal_endpoint_messages_send(), portal_get_upload_attachment(), requeue(), sess_number(), sess_ref_add(), sess_ref_dec(), sess_ref_stamp(), sess_ref_total(), sess_refresh_check(), sess_refresh_flush(), sess_refresh_stamp(), sess_trigger(), signal_refresh(), spool_check(), spool_check_file(), spool_error_stats(), spool_mktemp(), spool_start(), ssl_locking_callback(), stacker_nodes(), stacker_pop(), stacker_push(), statistics_refresh(), stats_adjust_by_name(), stats_adjust_by_num(), stats_decrement_by_name(), stats_decrement_by_num(), stats_get_name(), stats_get_name_pos(), stats_get_value_by_name(), stats_get_value_by_num(), stats_increment_by_name(), stats_increment_by_num(), stats_set_by_name(), stats_set_by_num(), and tank_cycle().

5.82.1.4 int mutex_unlock (pthread_mutex_t * lock)

Unlock a pthread mutex.

See also:

pthread_mutex_unlock()

Parameters:

lock a pointer to the mutex to be unlocked.

Returns:

0 on success, or an error number on failure.

Definition at line 57 of file mutex.c.

References log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, and MEMORYBUF.

Referenced by `con_decrement_refs()`, `con_increment_refs()`, `con_reverse_check()`, `con_reverse_domain()`, `con_reverse_enqueue()`, `con_reverse_status()`, `dequeue()`, `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_params_generate()`, `log_disable()`, `log_enable()`, `log_internal()`, `meta_user_ref_add()`, `meta_user_ref_dec()`, `meta_user_ref_protocol_total()`, `meta_user_ref_stamp()`, `meta_user_ref_total()`, `mm_sec_alloc()`, `mm_sec_free()`, `mm_sec_stats()`, `pattern_check()`, `pattern_stop()`, `pattern_update()`, `pool_get_failures()`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `pool_set_obj()`, `pool_swap_obj()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_send()`, `portal_get_upload_attachment()`, `requeue()`, `sess_number()`, `sess_ref_add()`, `sess_ref_dec()`, `sess_ref_stamp()`, `sess_ref_total()`, `sess_refresh_check()`, `sess_refresh_flush()`, `sess_refresh_stamp()`, `sess_trigger()`, `signal_refresh()`, `spool_check()`, `spool_check_file()`, `spool_error_stats()`, `spool_mktemp()`, `spool_start()`, `ssl_locking_callback()`, `stacker_nodes()`, `stacker_pop()`, `stacker_push()`, `statistics_refresh()`, `stats_adjust_by_name()`, `stats_adjust_by_num()`, `stats_decrement_by_name()`, `stats_decrement_by_num()`, `stats_get_name()`, `stats_get_name_pos()`, `stats_get_value_by_name()`, `stats_get_value_by_num()`, `stats_increment_by_name()`, `stats_increment_by_num()`, `stats_set_by_name()`, `stats_set_by_num()`, and `tank_cycle()`.

5.83 src/core/thread/rwlock.c File Reference

```
#include "magma.h"
```

Functions

- int [rwlock_init](#) (pthread_rwlock_t *[lock](#), pthread_rwlockattr_t *attr)
Initialize a pthread read/write lock.
- int [rwlock_lock_write](#) (pthread_rwlock_t *[lock](#))
Attempt to acquire a pthread read/write lock for writing, blocking if necessary.
- int [rwlock_lock_read](#) (pthread_rwlock_t *[lock](#))
Attempt to acquire a pthread read/write lock for reading, blocking if necessary.
- int [rwlock_unlock](#) (pthread_rwlock_t *[lock](#))
Unlock a pthread read/write lock.
- int [rwlock_destroy](#) (pthread_rwlock_t *[lock](#))
Free a pthread read/write lock and free its resources.
- int [rwlock_attr_init](#) (pthread_rwlockattr_t *attr)
Initialize a pthread read/write lock attributes object with default values.
- int [rwlock_attr_destroy](#) (pthread_rwlockattr_t *attr)
Free a pthread read/write lock attributes object.
- int [rwlock_attr_setkind](#) (pthread_rwlockattr_t *attr, int pref)
Set the kind of a pthread read/write lock attributes object.
- int [rwlock_attr_getkind](#) (pthread_rwlockattr_t *attr, int *pref)
Get the kind of a pthread read/write lock attributes object.

5.83.1 Function Documentation

5.83.1.1 int [rwlock_attr_destroy](#) (pthread_rwlockattr_t * *attr*)

Free a pthread read/write lock attributes object. [rwlock.c](#)

See also:

`pthread_rwlockattr_destroy()`

Parameters:

attr a pointer to the read/write lock attributes object to be freed.

Returns:

0 on success or an error number on failure.

Definition at line 121 of file `rwlock.c`.

References `log_pedantic`.

Referenced by `meta_alloc()`.

5.83.1.2 `int rwlock_attr_getkind(pthread_rwlockattr_t * attr, int * pref)`

Get the kind of a pthread read/write lock attributes object.

See also:

`pthread_rwlockattr_getkind_np()`

Parameters:

attr a pointer to the read/write lock attributes object to be queried.

pref a pointer to an integer that will receive the kind of the read/write lock.

Returns:

0 on success or an error number on failure.

Definition at line 155 of file `rwlock.c`.

References `log_pedantic`.

5.83.1.3 `int rwlock_attr_init(pthread_rwlockattr_t * attr)`

Initialize a pthread read/write lock attributes object with default values.

See also:

`pthread_rwlockattr_init()`

Parameters:

attr a pointer to the read/write lock attributes object to be initialized.

Returns:

0 on success or an error number on failure.

Definition at line 105 of file `rwlock.c`.

References `log_pedantic`.

Referenced by `meta_alloc()`.

5.83.1.4 `int rwlock_attr_setkind(pthread_rwlockattr_t * attr, int pref)`

Set the kind of a pthread read/write lock attributes object.

See also:

`pthread_rwlockattr_setkind_np()`

Parameters:

attr a pointer to the read/write lock attributes object to be adjusted.

pref the kind of the read/write lock: `PTHREAD_RWLOCK_PREFER_READER_NP`, `PTHREAD_RWLOCK_PREFER_WRITER_NP`, or `PTHREAD_RWLOCK_PREFER_WRITER_NONRECURSIVE_NP`.

Returns:

0 on success or an error number on failure.

Definition at line 138 of file rwlock.c.

References log_pedantic.

Referenced by meta_alloc().

5.83.1.5 int **rwlock_destroy** (pthread_rwlock_t * *lock*)

Free a pthread read/write lock and free its resources.

See also:

pthread_rwlock_destroy()

Parameters:

lock a pointer to the read/write lock to be destroyed.

Returns:

0 on success or an error number on failure.

Definition at line 89 of file rwlock.c.

References log_pedantic.

Referenced by inx_free(), magma_folder_free(), message_free(), meta_alloc(), and meta_free().

5.83.1.6 int **rwlock_init** (pthread_rwlock_t * *lock*, pthread_rwlockattr_t * *attr*)

Initialize a pthread read/write lock.

See also:

pthread_rwlock_init()

Parameters:

lock a pointer to the read/write lock to be initialized.

attr an optional pointer to a set of lock attributes, or the default attributes if NULL is specified.

Returns:

0 on success or an error number on failure.

Definition at line 17 of file rwlock.c.

References log_pedantic.

Referenced by inx_alloc(), magma_folder_alloc(), message_alloc(), and meta_alloc().

5.83.1.7 int **rwlock_lock_read** (pthread_rwlock_t * *lock*)

Attempt to acquire a pthread read/write lock for reading, blocking if necessary.

See also:

pthread_rwlock_rdlock()

Parameters:

lock the read-write lock to be tried.

Returns:

0 on success or an error number on failure.

Definition at line 53 of file rwlock.c.

References log_pedantic.

Referenced by inx_auto_read(), inx_lock_read(), meta_user_rlock(), register_abuse_check_blocklist(), spool_mktemp(), status(), and status_get().

5.83.1.8 int rwlock_lock_write (pthread_rwlock_t * lock)

Attempt to acquire a pthread read/write lock for writing, blocking if necessary.

See also:

pthread_rwlock_wrlock()

Parameters:

lock a pointer to the read/write lock to be acquired.

Returns:

0 on success or an error number on failure.

Definition at line 35 of file rwlock.c.

References log_pedantic.

Referenced by inx_auto_write(), inx_lock_write(), meta_user_wlock(), register_blocklist_update(), spool_cleanup(), status_set(), and status_signal().

5.83.1.9 int rwlock_unlock (pthread_rwlock_t * lock)

Unlock a pthread read/write lock.

See also:

pthread_rwlock_unlock()

Parameters:

lock a pointer to the read/write lock to be unlocked.

Returns:

0 on success or an error number on failure.

Definition at line 71 of file rwlock.c.

References log_pedantic.

Referenced by inx_auto_unlock(), inx_unlock(), meta_user_unlock(), register_abuse_check_blocklist(), register_blocklist_update(), spool_cleanup(), spool_mktemp(), status(), status_get(), status_set(), and status_signal().

5.84 src/core/thread/thread.c File Reference

```
#include "magma.h"
```

Functions

- pthread_t [thread_get_thread_id](#) (void)
Get the id of the calling thread.
- int_t [thread_launch](#) (pthread_t *thread, void *function, void *data)
Launch a thread to execute a specified function.
- pthread_t * [thread_alloc](#) (void *function, void *data)
Launch a function in a freshly created thread.
- int_t [thread_join](#) (pthread_t thread)
Block until a specified thread finishes execution.
- int_t [thread_result](#) (pthread_t thread, void **result)
Block until a specified thread finishes execution and store its exit value.
- int_t [thread_signal](#) (pthread_t thread, int_t signal)
Send a specified signal to a thread.
- int_t [thread_cancel](#) (pthread_t thread)
Send a cancellation request to a thread.
- void [thread_cancel_enable](#) (void)
Set the calling thread to be cancelable.
- void [thread_cancel_disable](#) (void)
Set the calling thread to be not cancelable.

5.84.1 Function Documentation

5.84.1.1 pthread_t* [thread_alloc](#) (void *function, void *data)

Launch a function in a freshly created thread. thread.c

See also:

[pthread_create_new\(\)](#)

Parameters:

function a pointer to the function to be executed.
data a pointer to data to be passed to the executed function.

Returns:

NULL on failure, or a pointer to the newly created pthread on success.

Definition at line 62 of file thread.c.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_free\(\)](#), and [thread_launch\(\)](#).

Referenced by [net_listen\(\)](#).

5.84.1.2 `int_t thread_cancel (pthread_t thread)`

Send a cancellation request to a thread.

Parameters:

thread the pthread id of the thread to be canceled.

Returns:

0 on success or a non-zero error number on failure.

Definition at line 131 of file thread.c.

References `log_pedantic`.

5.84.1.3 `void thread_cancel_disable (void)`

Set the calling thread to be not cancelable.

Returns:

This function returns no value.

Definition at line 162 of file thread.c.

Referenced by `process_maint()`.

5.84.1.4 `void thread_cancel_enable (void)`

Set the calling thread to be cancelable.

Returns:

This function returns no value.

Definition at line 153 of file thread.c.

Referenced by `process_maint()`.

5.84.1.5 `pthread_t thread_get_thread_id (void)`

Get the id of the calling thread.

Returns:

the id of the calling thread.

Definition at line 14 of file thread.c.

Referenced by `rand_start()`, `rand_thread_start()`, `spool_mktemp()`, and `ssl_thread_id_callback()`.

5.84.1.6 `int_t thread_join (pthread_t thread)`

Block until a specified thread finishes execution.

Note:

`pthread_join()`

Parameters:

thread the thread to wait upon.

Returns:

0 on success, or an error number on failure.

Definition at line 86 of file thread.c.

References log_pedantic.

Referenced by net_listen(), process_stop(), queue_shutdown(), and signal_shutdown().

5.84.1.7 int_t thread_launch (pthread_t * *thread*, void * *function*, void * *data*)

Launch a thread to execute a specified function.

See also:

pthread_create()

Parameters:

thread a pointer to a pthread object to receive the new thread id.

function a pointer to a function to be executed inside the new thread.

data a pointer to be data to be passed in as input to the called function.

Returns:

0 on success, or a non-zero error code on failure.

Definition at line 27 of file thread.c.

References log_pedantic, magma, magma_t::system, and magma_t::thread_stack_size.

Referenced by process_start(), queue_init(), signal_shutdown(), and thread_alloc().

5.84.1.8 int_t thread_result (pthread_t *thread*, void ** *result*)

Block until a specified thread finishes execution and store its exit value.

Note:

pthread_join()

Parameters:

thread the thread to wait upon.

result a pointer to a block of memory to receive the target thread exit value.

Returns:

0 on success, or an error number on failure.

Definition at line 104 of file thread.c.

References log_pedantic.

5.84.1.9 `int_t thread_signal(pthread_t thread, int_t signal)`

Send a specified signal to a thread.

Parameters:

thread the target thread id.

signal the number of the signal to be delivered.

Returns:

0 on success or an error number on failure.

Definition at line 121 of file thread.c.

Referenced by `process_stop()`, and `queue_signal()`.

5.85 src/engine/context/thread.c File Reference

```
#include "magma.h"
```

Functions

- void [thread_stop](#) (void)
Prepare a thread to exit by destroying its MySQL and OpenSSL thread storage, and the thread specific mail cache.
- [bool_t thread_start](#) (void)
Setup a new thread by registering the signal handlers and preparing the MySQL thread specific storage.

5.85.1 Function Documentation

5.85.1.1 [bool_t thread_start](#) (void)

Setup a new thread by registering the signal handlers and preparing the MySQL thread specific storage. [thread.c](#)

Returns:

true on success or false on failure.

Definition at line 27 of file [thread.c](#).

References [signal_thread_start\(\)](#), and [sql_thread_start\(\)](#).

Referenced by [dequeue\(\)](#), [net_accept\(\)](#), and [process_maint\(\)](#).

5.85.1.2 [void thread_stop](#) (void)

Prepare a thread to exit by destroying its MySQL and OpenSSL thread storage, and the thread specific mail cache.

Returns:

This function returns no value.

Definition at line 14 of file [thread.c](#).

References [mail_cache_thread_stop\(\)](#), [sql_thread_stop\(\)](#), and [ssl_thread_stop\(\)](#).

Referenced by [dequeue\(\)](#), [net_accept\(\)](#), and [process_maint\(\)](#).

5.86 src/core/thread/thread.h File Reference

Functions

- pthread_t * [thread_alloc](#) (void *function, void *data)
thread.c
- int_t [thread_cancel](#) (pthread_t thread)
Send a cancellation request to a thread.
- void [thread_cancel_disable](#) (void)
Set the calling thread to be not cancelable.
- void [thread_cancel_enable](#) (void)
Set the calling thread to be cancelable.
- pthread_t [thread_get_thread_id](#) (void)
Get the id of the calling thread.
- int_t [thread_join](#) (pthread_t thread)
Block until a specified thread finishes execution.
- int_t [thread_launch](#) (pthread_t *thread, void *function, void *data)
Launch a thread to execute a specified function.
- int_t [thread_result](#) (pthread_t thread, void **result)
Block until a specified thread finishes execution and store its exit value.
- int_t [thread_signal](#) (pthread_t thread, int_t signal)
Send a specified signal to a thread.
- int [rwlock_attr_destroy](#) (pthread_rwlockattr_t *attr)
rwlock.c
- int [rwlock_attr_getkind](#) (pthread_rwlockattr_t *attr, int *pref)
Get the kind of a pthread read/write lock attributes object.
- int [rwlock_attr_init](#) (pthread_rwlockattr_t *attr)
Initialize a pthread read/write lock attributes object with default values.
- int [rwlock_attr_setkind](#) (pthread_rwlockattr_t *attr, int pref)
Set the kind of a pthread read/write lock attributes object.
- int [rwlock_destroy](#) (pthread_rwlock_t *lock)
Free a pthread read/write lock and free its resources.
- int [rwlock_init](#) (pthread_rwlock_t *lock, pthread_rwlockattr_t *attr)
Initialize a pthread read/write lock.
- int [rwlock_lock_read](#) (pthread_rwlock_t *lock)
Attempt to acquire a pthread read/write lock for reading, blocking if necessary.
- int [rwlock_lock_write](#) (pthread_rwlock_t *lock)

Attempt to acquire a pthread read/write lock for writing, blocking if necessary.

- int [rwlock_unlock](#) (pthread_rwlock_t *[lock](#))

Unlock a pthread read/write lock.

- void * [tkey_get](#) (pthread_key_t key)

keys.c

- int [tkey_init](#) (pthread_key_t *key, void(*destructor)(void *))

Create a thread-specific data key.

- int [tkey_set](#) (pthread_key_t key, void *value)

Set the calling thread's value for a pthread key.

- int [mutex_destroy](#) (pthread_mutex_t *[lock](#))

mutex.c

- int [mutex_init](#) (pthread_mutex_t *[lock](#), pthread_mutexattr_t *attr)

Initialize a pthread mutex with the given attributes.

- int [mutex_lock](#) (pthread_mutex_t *[lock](#))

Acquire a pthread mutex, blocking if necessary.

- int [mutex_unlock](#) (pthread_mutex_t *[lock](#))

Unlock a pthread mutex.

5.86.1 Function Documentation

5.86.1.1 int [mutex_destroy](#) (pthread_mutex_t * *lock*)

[mutex.c](#) [mutex.c](#)

See also:

[pthread_mutex_destroy\(\)](#)

Parameters:

lock a pointer to the mutex to be freed.

Returns:

0 on success, or an error number on failure.

Definition at line 77 of file [mutex.c](#).

References [log_pedantic](#), and [MEMORYBUF](#).

Referenced by [con_destroy\(\)](#), [meta_free\(\)](#), [pool_free\(\)](#), [protocol_secure\(\)](#), [queue_shutdown\(\)](#), [sess_destroy\(\)](#), [ssl_stop\(\)](#), [stacker_free\(\)](#), and [stats_shutdown\(\)](#).

5.86.1.2 int mutex_init (pthread_mutex_t * *lock*, pthread_mutexattr_t * *attr*)

Initialize a pthread mutex with the given attributes.

See also:

pthread_mutex_init()

Parameters:

lock a pointer to a mutex that will be initialized.

attr a pointer to a mutex attributes holder, or NULL to use system default values.

Returns:

0 on success, or an error number on failure.

Definition at line 17 of file mutex.c.

References log_pedantic, and MEMORYBUF.

Referenced by con_init(), meta_alloc(), pool_alloc(), queue_init(), ssl_start(), stacker_alloc(), and stats_init().

5.86.1.3 int mutex_lock (pthread_mutex_t * *lock*)

Acquire a pthread mutex, blocking if necessary.

See also:

pthread_mutex_lock()

Parameters:

lock a pointer to the mutex to be locked.

Returns:

0 on success, or an error number on failure.

Definition at line 37 of file mutex.c.

References log_options, M_LOG_PEDANTIC, M_LOG_STACK_TRACE, and MEMORYBUF.

Referenced by con_decrement_refs(), con_increment_refs(), con_reverse_check(), con_reverse_domain(), con_reverse_enqueue(), con_reverse_status(), dequeue(), dh_exchange_2048(), dh_exchange_4096(), dh_params_generate(), log_disable(), log_enable(), log_internal(), meta_user_ref_add(), meta_user_ref_dec(), meta_user_ref_protocol_total(), meta_user_ref_stamp(), meta_user_ref_total(), mm_sec_alloc(), mm_sec_free(), mm_sec_stats(), pattern_check(), pattern_stop(), pattern_update(), pool_get_failures(), pool_get_obj(), pool_pull(), pool_release(), pool_set_obj(), pool_swap_obj(), portal_endpoint_attachments_add(), portal_endpoint_attachments_remove(), portal_endpoint_messages_compose(), portal_endpoint_messages_send(), portal_get_upload_attachment(), requeue(), sess_number(), sess_ref_add(), sess_ref_dec(), sess_ref_stamp(), sess_ref_total(), sess_refresh_check(), sess_refresh_flush(), sess_refresh_stamp(), sess_trigger(), signal_refresh(), spool_check(), spool_check_file(), spool_error_stats(), spool_mktemp(), spool_start(), ssl_locking_callback(), stacker_nodes(), stacker_pop(), stacker_push(), statistics_refresh(), stats_adjust_by_name(), stats_adjust_by_num(), stats_decrement_by_name(), stats_decrement_by_num(), stats_get_name(), stats_get_name_pos(), stats_get_value_by_name(), stats_get_value_by_num(), stats_increment_by_name(), stats_increment_by_num(), stats_set_by_name(), stats_set_by_num(), and tank_cycle().

5.86.1.4 int mutex_unlock (pthread_mutex_t * *lock*)

Unlock a pthread mutex.

See also:

`pthread_mutex_unlock()`

Parameters:

lock a pointer to the mutex to be unlocked.

Returns:

0 on success, or an error number on failure.

Definition at line 57 of file `mutex.c`.

References `log_options`, `M_LOG_PEDANTIC`, `M_LOG_STACK_TRACE`, and `MEMORYBUF`.

Referenced by `con_decrement_refs()`, `con_increment_refs()`, `con_reverse_check()`, `con_reverse_domain()`, `con_reverse_enqueue()`, `con_reverse_status()`, `dequeue()`, `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_params_generate()`, `log_disable()`, `log_enable()`, `log_internal()`, `meta_user_ref_add()`, `meta_user_ref_dec()`, `meta_user_ref_protocol_total()`, `meta_user_ref_stamp()`, `meta_user_ref_total()`, `mm_sec_alloc()`, `mm_sec_free()`, `mm_sec_stats()`, `pattern_check()`, `pattern_stop()`, `pattern_update()`, `pool_get_failures()`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `pool_set_obj()`, `pool_swap_obj()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_send()`, `portal_get_upload_attachment()`, `requeue()`, `sess_number()`, `sess_ref_add()`, `sess_ref_dec()`, `sess_ref_stamp()`, `sess_ref_total()`, `sess_refresh_check()`, `sess_refresh_flush()`, `sess_refresh_stamp()`, `sess_trigger()`, `signal_refresh()`, `spool_check()`, `spool_check_file()`, `spool_error_stats()`, `spool_mkdir()`, `spool_start()`, `ssl_locking_callback()`, `stacker_nodes()`, `stacker_pop()`, `stacker_push()`, `statistics_refresh()`, `stats_adjust_by_name()`, `stats_adjust_by_num()`, `stats_decrement_by_name()`, `stats_decrement_by_num()`, `stats_get_name()`, `stats_get_name_pos()`, `stats_get_value_by_name()`, `stats_get_value_by_num()`, `stats_increment_by_name()`, `stats_increment_by_num()`, `stats_set_by_name()`, `stats_set_by_num()`, and `tank_cycle()`.

5.86.1.5 int rwlock_attr_destroy (pthread_rwlockattr_t * attr)

[rwlock.c](#) [rwlock.c](#)

See also:

`pthread_rwlockattr_destroy()`

Parameters:

attr a pointer to the read/write lock attributes object to be freed.

Returns:

0 on success or an error number on failure.

Definition at line 121 of file `rwlock.c`.

References `log_pedantic`.

Referenced by `meta_alloc()`.

5.86.1.6 int rwlock_attr_getkind (pthread_rwlockattr_t * attr, int * pref)

Get the kind of a pthread read/write lock attributes object.

See also:

`pthread_rwlockattr_getkind_np()`

Parameters:

attr a pointer to the read/write lock attributes object to be queried.

pref a pointer to an integer that will receive the kind of the read/write lock.

Returns:

0 on success or an error number on failure.

Definition at line 155 of file rwlock.c.

References log_pedantic.

5.86.1.7 int rwlock_attr_init (pthread_rwlockattr_t * attr)

Initialize a pthread read/write lock attributes object with default values.

See also:

pthread_rwlockattr_init()

Parameters:

attr a pointer to the read/write lock attributes object to be initialized.

Returns:

0 on success or an error number on failure.

Definition at line 105 of file rwlock.c.

References log_pedantic.

Referenced by meta_alloc().

5.86.1.8 int rwlock_attr_setkind (pthread_rwlockattr_t * attr, int pref)

Set the kind of a pthread read/write lock attributes object.

See also:

pthread_rwlockattr_setkind_np()

Parameters:

attr a pointer to the read/write lock attributes object to be adjusted.

pref the kind of the read/write lock: PTHREAD_RWLOCK_PREFER_READER_NP, PTHREAD_RWLOCK_PREFER_WRITER_NP, or PTHREAD_RWLOCK_PREFER_WRITER_NONRECURSIVE_NP.

Returns:

0 on success or an error number on failure.

Definition at line 138 of file rwlock.c.

References log_pedantic.

Referenced by meta_alloc().

5.86.1.9 int rwlock_destroy (pthread_rwlock_t * lock)

Free a pthread read/write lock and free its resources.

See also:

`pthread_rwlock_destroy()`

Parameters:

lock a pointer to the read/write lock to be destroyed.

Returns:

0 on success or an error number on failure.

Definition at line 89 of file `rwlock.c`.

References `log_pedantic`.

Referenced by `inx_free()`, `magma_folder_free()`, `message_free()`, `meta_alloc()`, and `meta_free()`.

5.86.1.10 `int pthread_rwlock_init(pthread_rwlock_t *lock, pthread_rwlockattr_t *attr)`

Initialize a pthread read/write lock.

See also:

`pthread_rwlock_init()`

Parameters:

lock a pointer to the read/write lock to be initialized.

attr an optional pointer to a set of lock attributes, or the default attributes if NULL is specified.

Returns:

0 on success or an error number on failure.

Definition at line 17 of file `rwlock.c`.

References `log_pedantic`.

Referenced by `inx_alloc()`, `magma_folder_alloc()`, `message_alloc()`, and `meta_alloc()`.

5.86.1.11 `int pthread_rwlock_read(pthread_rwlock_t *lock)`

Attempt to acquire a pthread read/write lock for reading, blocking if necessary.

See also:

`pthread_rwlock_rdlock()`

Parameters:

lock the read-write lock to be tried.

Returns:

0 on success or an error number on failure.

Definition at line 53 of file `rwlock.c`.

References `log_pedantic`.

Referenced by `inx_auto_read()`, `inx_lock_read()`, `meta_user_rlock()`, `register_abuse_check_blocklist()`, `spool_mktemp()`, `status()`, and `status_get()`.

5.86.1.12 `int rwlock_lock_write(pthread_rwlock_t * lock)`

Attempt to acquire a pthread read/write lock for writing, blocking if necessary.

See also:

`pthread_rwlock_wrlock()`

Parameters:

lock a pointer to the read/write lock to be acquired.

Returns:

0 on success or an error number on failure.

Definition at line 35 of file `rwlock.c`.

References `log_pedantic`.

Referenced by `inx_auto_write()`, `inx_lock_write()`, `meta_user_wlock()`, `register_blocklist_update()`, `spool_cleanup()`, `status_set()`, and `status_signal()`.

5.86.1.13 `int rwlock_unlock(pthread_rwlock_t * lock)`

Unlock a pthread read/write lock.

See also:

`pthread_rwlock_unlock()`

Parameters:

lock a pointer to the read/write lock to be unlocked.

Returns:

0 on success or an error number on failure.

Definition at line 71 of file `rwlock.c`.

References `log_pedantic`.

Referenced by `inx_auto_unlock()`, `inx_unlock()`, `meta_user_unlock()`, `register_abuse_check_blocklist()`, `register_blocklist_update()`, `spool_cleanup()`, `spool_mktemp()`, `status()`, `status_get()`, `status_set()`, and `status_signal()`.

5.86.1.14 `pthread_t* thread_alloc(void * function, void * data)`

`thread.c` `thread.c`

See also:

`pthread_create_new()`

Parameters:

function a pointer to the function to be executed.

data a pointer to data to be passed to the executed function.

Returns:

NULL on failure, or a pointer to the newly created pthread on success.

Definition at line 62 of file thread.c.

References log_pedantic, mm_alloc(), mm_free(), and thread_launch().

Referenced by net_listen().

5.86.1.15 int_t thread_cancel (pthread_t *thread*)

Send a cancellation request to a thread.

Parameters:

thread the pthread id of the thread to be canceled.

Returns:

0 on success or a non-zero error number on failure.

Definition at line 131 of file thread.c.

References log_pedantic.

5.86.1.16 void thread_cancel_disable (void)

Set the calling thread to be not cancelable.

Returns:

This function returns no value.

Definition at line 162 of file thread.c.

Referenced by process_maint().

5.86.1.17 void thread_cancel_enable (void)

Set the calling thread to be cancelable.

Returns:

This function returns no value.

Definition at line 153 of file thread.c.

Referenced by process_maint().

5.86.1.18 pthread_t thread_get_thread_id (void)

Get the id of the calling thread.

Returns:

the id of the calling thread.

Definition at line 14 of file thread.c.

Referenced by rand_start(), rand_thread_start(), spool_mktemp(), and ssl_thread_id_callback().

5.86.1.19 int_t thread_join (pthread_t *thread*)

Block until a specified thread finishes execution.

Note:

pthread_join()

Parameters:

thread the thread to wait upon.

Returns:

0 on success, or an error number on failure.

Definition at line 86 of file thread.c.

References log_pedantic.

Referenced by net_listen(), process_stop(), queue_shutdown(), and signal_shutdown().

5.86.1.20 int_t thread_launch (pthread_t * *thread*, void * *function*, void * *data*)

Launch a thread to execute a specified function.

See also:

pthread_create()

Parameters:

thread a pointer to a pthread object to receive the new thread id.

function a pointer to a function to be executed inside the new thread.

data a pointer to be data to be passed in as input to the called function.

Returns:

0 on success, or a non-zero error code on failure.

Definition at line 27 of file thread.c.

References log_pedantic, magma, magma_t::system, and magma_t::thread_stack_size.

Referenced by process_start(), queue_init(), signal_shutdown(), and thread_alloc().

5.86.1.21 int_t thread_result (pthread_t *thread*, void ** *result*)

Block until a specified thread finishes execution and store its exit value.

Note:

pthread_join()

Parameters:

thread the thread to wait upon.

result a pointer to a block of memory to receive the target thread exit value.

Returns:

0 on success, or an error number on failure.

Definition at line 104 of file thread.c.

References log_pedantic.

5.86.1.22 int_t thread_signal (pthread_t *thread*, int_t *signal*)

Send a specified signal to a thread.

Parameters:

thread the target thread id.

signal the number of the signal to be delivered.

Returns:

0 on success or an error number on failure.

Definition at line 121 of file thread.c.

Referenced by process_stop(), and queue_signal().

5.86.1.23 void* tkey_get (pthread_key_t *key*)

keys.c keys.c

See also:

pthread_getspecific()

Parameters:

key the pthread key to be queried.

Returns:

NULL on failure or a pointer to the key's thread-specific data value on success.

Definition at line 35 of file keys.c.

5.86.1.24 int tkey_init (pthread_key_t * *key*, void(*)(void *) *destructor*)

Create a thread-specific data key.

See also:

pthread_key_create()

Parameters:

key a pointer to pthread key to be initialized for thread-local storage.

destructor if not NULL, an optional function pointer to be called at thread exit to release the data.

Returns:

0 on success, or an error number on failure.

Definition at line 17 of file keys.c.

References log_pedantic.

5.86.1.25 `int tkey_set(pthread_key_t key, void * value)`

Set the calling thread's value for a pthread key.

See also:

`pthread_setspecific()`

Parameters:

key the pthread key to have its thread-local value modified.

value a pointer to the new thread-specific value of the specified key.

Returns:

0 on success, or an error number on failure.

Definition at line 46 of file keys.c.

References `log_pedantic`.

5.87 src/core/type.c File Reference

```
#include "magma.h"
```

Functions

- char * [type](#) ([M_TYPE](#) type)

5.87.1 Function Documentation

5.87.1.1 char* type (M_TYPE type)

Takes a type code and returns the fully enumerated string associated with that type.

Parameters:

type The type code to evaluate.

Returns:

Null terminated string containing type name. String is stored in static buffer and returned as a pointer.

Definition at line 17 of file type.c.

References [M_TYPE_BLOCK](#), [M_TYPE_BOOLEAN](#), [M_TYPE_DOUBLE](#), [M_TYPE_ENUM](#), [M_TYPE_FLOAT](#), [M_TYPE_INT16](#), [M_TYPE_INT32](#), [M_TYPE_INT64](#), [M_TYPE_INT8](#), [M_TYPE_MULTI](#), [M_TYPE_NULLER](#), [M_TYPE_PLACER](#), [M_TYPE_STRINGER](#), [M_TYPE_UINT16](#), [M_TYPE_UINT32](#), [M_TYPE_UINT64](#), and [M_TYPE_UINT8](#).

Referenced by [__attribute__\(\)](#), [aes_artifact_decrypt\(\)](#), [aes_artifact_encrypt\(\)](#), [ar_dupe\(\)](#), [ar_free\(\)](#), [cache_free\(\)](#), [cache_output_settings\(\)](#), [cache_set_value\(\)](#), [cache_validate\(\)](#), [chunk_buffer_size\(\)](#), [chunk_header_size\(\)](#), [chunk_header_type\(\)](#), [config_free\(\)](#), [encrypted_chunk_get\(\)](#), [imap_parse_array\(\)](#), [imap_parse_dataitems\(\)](#), [mail_modify_part\(\)](#), [meta_data_delete_folder\(\)](#), [meta_data_fetch_folders\(\)](#), [meta_data_insert_folder\(\)](#), [meta_data_update_folder_name\(\)](#), [mt_dupe\(\)](#), [mt_get_char\(\)](#), [mt_get_length\(\)](#), [mt_get_number\(\)](#), [mt_get_type\(\)](#), [mt_is_empty\(\)](#), [mt_is_number\(\)](#), [mt_set_type\(\)](#), [naked_message_get\(\)](#), [naked_message_set\(\)](#), [part_decrypt\(\)](#), [portal_message_attachments\(\)](#), [portal_message_body\(\)](#), [prime_key_decrypt\(\)](#), [prime_pem_unwrap\(\)](#), [prime_pem_wrap\(\)](#), [prime_set\(\)](#), [prime_unpack\(\)](#), [prime_unpack_fields\(\)](#), [prime_unpack_validate\(\)](#), [relay_free\(\)](#), [relay_output_settings\(\)](#), [relay_set_value\(\)](#), [relay_validate\(\)](#), [servers_free\(\)](#), [servers_get_by_protocol\(\)](#), [servers_output_settings\(\)](#), [servers_set_value\(\)](#), [servers_validate\(\)](#), [signature_full_verify\(\)](#), and [signature_tree_get\(\)](#).

5.88 src/engine/config/cache/cache.c File Reference

```
#include "magma.h"
#include "keys.h"
```

Functions

- void `cache_free` (void)
Free magma's cache server configuration options.
- `cache_t * cache_alloc` (uint32_t *number*)
Allocate a new cache server configuration entry and initialize it to its default values.
- `bool_t cache_validate` (void)
Validate all of the configured magma cache server instances.
- void `cache_output_settings` (void)
Log the full contents of the magma cache configuration settings.
- `bool_t cache_set_value` (`cache_keys_t` **setting*, `cache_t` **cache*, `stringer_t` **value*)
Set the value of a key for a specified cache server configuration entry.
- `bool_t cache_config` (`stringer_t` **name*, `stringer_t` **value*)
Set the value of a cache server configuration entry by name.
- void `cache_output_help` (void)
Log all cache key information to be returned via "magma -h".

5.88.1 Function Documentation

5.88.1.1 `cache_t* cache_alloc (uint32_t number)`

Allocate a new cache server configuration entry and initialize it to its default values.

Parameters:

number the index of the cache server instance in the global cache server array.

Returns:

NULL on failure, or a pointer to the requested cache server configuration entry on success.

Definition at line 64 of file cache.c.

References `magma_t::cache`, `cache_keys`, `cache_set_value()`, `magma_t::iface`, `log_critical`, `magma`, and `mm_alloc()`.

Referenced by `cache_config()`.

5.88.1.2 `bool_t cache_config (stringer_t * name, stringer_t * value)`

Set the value of a cache server configuration entry by name.

Note:

This function is designed to operate on lines from a configuration source, like a file or database. When a named server is referenced for the first time, a new cache server configuration instance will be allocated.

Parameters:

name a managed string containing the human-readable cache server configuration parameter to be set.

value a managed string containing the new value of the cache server config key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 425 of file cache.c.

References bracket_extract_pl(), cache_alloc(), cache_keys, cache_set_value(), CONSTANT, log_critical, MAGMA_CACHE_INSTANCES, NULLER, pl_char_get(), pl_empty(), pl_length_get(), PLACER, placer_t, st_char_get(), st_cmp_ci_eq(), st_cmp_ci_starts(), st_length_get(), st_length_int(), and uint32_conv_bl().

Referenced by config_load_cmdline_settings(), config_load_database_settings(), and config_load_file_settings().

5.88.1.3 void cache_free (void)

Free magma's cache server configuration options.

Note:

This function assumes all worker threads have finished, and should only be called during the normal shutdown process.

Parameters:

This function returns no value.

Definition at line 17 of file cache.c.

References magma_t::cache, cache_keys, magma_t::iface, log_pedantic, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_ENUM, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_CACHE_INSTANCES, mm_free(), cache_keys_t::norm, ns_empty(), ns_free(), cache_keys_t::offset, st_empty, st_free(), type(), and multi_t::type.

Referenced by config_free().

5.88.1.4 void cache_output_help (void)

Log all cache key information to be returned via "magma -h".

Returns:

This function returns no value.

Definition at line 484 of file cache.c.

References multi_t::bl, cache_keys, config_output_value_generic(), log_info, log_options, M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE_DISABLE, M_LOG_TIME_DISABLE, cache_keys_t::norm, cache_keys_t::required, and multi_t::val.

Referenced by config_output_help().

5.88.1.5 void cache_output_settings (void)

Log the full contents of the magma cache configuration settings.

Note:

This logs all the details of each memcached instance configured to work with magma.

Returns:

This function returns no value.

Definition at line 207 of file cache.c.

References magma_t::cache, cache_keys, magma_t::iface, log_info, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_CACHE_INSTANCES, cache_keys_t::norm, ns_empty(), cache_keys_t::offset, st_char_get(), st_empty, st_length_int(), multi_t::type, and type().

Referenced by config_output_settings().

5.88.1.6 bool_t cache_set_value (cache_keys_t * setting, cache_t * cache, stringer_t * value)

Set the value of a key for a specified cache server configuration entry.

Note:

This function will allocate space for a copy of the key value, and return an error if it is not able to convert it to the proper key data type.

Parameters:

setting a pointer to the cache server key to be set.

cache a pointer to the cache server configuration entry to be modified.

value a managed string containing the new value of the key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 282 of file cache.c.

References multi_t::binary, CONSTANT, CONTIGUOUS, HEAP, multi_t::i32, multi_t::i64, multi_t::i8, int32_conv_st(), int64_conv_st(), int8_conv_st(), log_critical, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, MANAGED_T, cache_keys_t::name, cache_keys_t::norm, multi_t::ns, ns_dupe(), ns_empty(), ns_free(), ns_import(), cache_keys_t::offset, multi_t::st, st_char_get(), st_cmp_ci_eq(), st_dupe_opts(), st_empty, st_free(), st_length_get(), type(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, uint16_conv_st(), uint32_conv_st(), uint64_conv_st(), uint8_conv_st(), and multi_t::val.

Referenced by cache_alloc(), and cache_config().

5.88.1.7 bool_t cache_validate (void)

Validate all of the configured magma cache server instances.

Note:

This makes set that all required config keys have been set, and that at least one host has been configured.

Returns:

true if all cache servers have been validated, or false on failure.

Definition at line 93 of file cache.c.

References magma_t::cache, cache_keys, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, magma_t::iface, log_critical, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_CACHE_INSTANCES, cache_keys_t::norm, ns_empty(), cache_keys_t::offset, st_empty, multi_t::type, type(), multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by config_validate_settings().

5.89 src/objects/mail/cache.c File Reference

```
#include "magma.h"
```

Functions

- void [mail_cache_destroy](#) (void *holder)
Free a cached mail message.
- [bool_t mail_cache_start](#) (void)
Initialize the thread-specific data key used for the mail message cache.
- void [mail_cache_stop](#) (void)
Destroy the thread-specific data key used for the mail message cache.
- void [mail_cache_thread_stop](#) (void)
Free the thread-specific data for the mail message cache.
- [stringer_t * mail_cache_get](#) (uint64_t messagenum)
Attempt to retrieve the contents of a message from the thread's cached data.
- void [mail_cache_reset](#) (void)
Reset the thread's mail cache and free any held message.
- void [mail_cache_set](#) (uint64_t messagenum, [stringer_t](#) *text)
Set the contents of the thread's mail cache.

5.89.1 Function Documentation

5.89.1.1 void mail_cache_destroy (void * holder)

Free a cached mail message. cache.c

Parameters:

holder a pointer to the cached mail message object to be freed.

Returns:

This function returns no value.

Definition at line 17 of file cache.c.

References [mm_free\(\)](#), [st_cleanup](#), and [mail_cache_t::text](#).

Referenced by [mail_cache_start\(\)](#), and [mail_cache_thread_stop\(\)](#).

5.89.1.2 stringer_t* mail_cache_get (uint64_t messagenum)

Attempt to retrieve the contents of a message from the thread's cached data.

Note:

The thread's cache only has the capacity to store a single message.

Parameters:

messagenum the id of the message to be retrieved.

Returns:

NULL on failure or a managed string containing the message data on success.

Definition at line 78 of file cache.c.

References mail_cache_t::messagenum, st_dup(), and mail_cache_t::text.

Referenced by mail_load_message().

5.89.1.3 void mail_cache_reset (void)

Reset the thread's mail cache and free any held message.

Returns:

This function returns no value.

Definition at line 97 of file cache.c.

References mail_cache_t::messagenum, mm_free(), st_cleanup, and mail_cache_t::text.

Referenced by imap_session_destroy(), and pop_session_destroy().

5.89.1.4 void mail_cache_set (uint64_t messagenum, stringer_t * text)

Set the contents of the thread's mail cache.

Parameters:

messagenum the numerical id of the message to be cached. a managed string containing the contents of the specified message to be cached.

Returns:

This function returns no value.

Definition at line 118 of file cache.c.

References CONTIGUOUS, HEAP, log_pedantic, MANAGED_T, mail_cache_t::messagenum, mm_alloc(), mm_free(), st_cleanup, st_dup_opts(), and mail_cache_t::text.

Referenced by mail_load_message().

5.89.1.5 bool_t mail_cache_start (void)

Initialize the thread-specific data key used for the mail message cache.

Returns:

true on success or false on failure.

Definition at line 33 of file cache.c.

References log_pedantic, and mail_cache_destroy().

Referenced by process_start().

5.89.1.6 void mail_cache_stop (void)

Destroy the thread-specific data key used for the mail message cache.

Returns:

This function returns no value.

Definition at line 47 of file cache.c.

References log_pedantic.

Referenced by process_stop().

5.89.1.7 void mail_cache_thread_stop (void)

Free the thread-specific data for the mail message cache.

Returns:

This function returns no value.

Definition at line 60 of file cache.c.

References mail_cache_destroy().

Referenced by thread_stop().

5.90 src/providers/consumers/cache.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t lib_load_cache` (void)
Initialize the memcached library and bind dynamically to the exported functions that are required.
- `const char * lib_version_cache` (void)
Return the version string of the memcached library.
- `bool_t cache_start` (void)
- `void cache_stop` (void)
Destroy the memcached client pool and all active connections.
- `void cache_flush` (void)
- `stringer_t * cache_get` (stringer_t *key)
Retrieve data from memcached by key.
- `uint64_t cache_get_u64` (stringer_t *key)
Retrieve a 64 bit value from memcached by key.
- `int_t cache_set` (stringer_t *key, stringer_t *object, time_t expiration)
Set a value in memcached by key.
- `int_t cache_set_u64` (stringer_t *key, uint64_t value, time_t expiration)
Set a 64 bit value in memcached by key.
- `int_t cache_add` (stringer_t *key, stringer_t *object, time_t expiration)
Add a new key/value pair to the memcached server.
- `int_t cache_silent_add` (stringer_t *key, stringer_t *object, time_t expiration)
Add a new key/value pair to the memcached server; but suppress any error messages.
- `int_t cache_append` (stringer_t *key, stringer_t *object, time_t expiration)
Append data to the value of a cached key on a memcached server.
- `int_t cache_delete` (stringer_t *key)
Delete a key from memcached immediately.
- `uint64_t cache_increment` (stringer_t *key, uint64_t offset, uint64_t initial, time_t expiration)
Increment the value stored with a key by memcached by a specified offset.
- `uint64_t cache_decrement` (stringer_t *key, uint64_t offset, uint64_t initial, time_t expiration)
Decrement the value stored with a key inside memcached, using the specified offset.

Variables

- `pool_t * cache_pool` = NULL

5.90.1 Function Documentation

5.90.1.1 `int_t cache_add (stringer_t * key, stringer_t * object, time_t expiration)`

Add a new key/value pair to the memcached server. `cache.c`

Parameters:

key a managed string with the name of the key to be added to the cache.

object a managed string containing the initial value of the new key.

expiration an expiration time, in seconds, for the newly cached data.

Definition at line 278 of file `cache.c`.

References `cache_pool`, `log_info`, `memcached_add_d`, `memcached_strerror_d`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

5.90.1.2 `int_t cache_append (stringer_t * key, stringer_t * object, time_t expiration)`

Append data to the value of a cached key on a memcached server.

Parameters:

key a managed string with the name of the target memcached key.

object a managed string containing the value of the data being appended to the original key.

expiration an expiration time, in seconds, for the modified cached data.

Definition at line 323 of file `cache.c`.

References `cache_pool`, `log_info`, `memcached_add_d`, `memcached_append_d`, `memcached_strerror_d`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

Referenced by `stamp_counter_increment()`.

5.90.1.3 `uint64_t cache_decrement (stringer_t * key, uint64_t offset, uint64_t initial, time_t expiration)`

Decrement the value stored with a key inside memcached, using the specified offset.

See also:

`memcached_decrement_with_initial()`

Note:

If the expiration valuse is `MEMCACHED_EXPIRATION_NOT_ADD` the key won't be added, but the initial value will be used as the return value.

Parameters:

key a managed string containing the name of the memcached key to be adjusted.

offset the offset by which the specified key's value should be decremented.

initial the initial value to seed the key with it does not already exist.

expiration the time, in seconds, when the cached key will expire.

Returns:

0 on failure, or the newly decremented value associated with the key, or the initial value, on success.

Definition at line 415 of file cache.c.

References `cache_pool`, `log_pedantic`, `memcached_decrement_with_initial_d`, `memcached_strerror_d`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

Referenced by `imap_login()`, `pop_pass()`, `smtp_auth_login()`, and `smtp_auth_plain()`.

5.90.1.4 `int_t cache_delete (stringer_t * key)`

Delete a key from memcached immediately.

Note:

`memcached_delete()`

Parameters:

key the name of the key to be deleted from the memcached server.

Returns:

0 on failure, or `MEMCACHED_SUCCESS` on success.

Definition at line 353 of file cache.c.

References `cache_pool`, `log_info`, `memcached_delete_d`, `memcached_strerror_d`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

Referenced by `lock_release()`.

5.90.1.5 `void cache_flush (void)`

Definition at line 124 of file cache.c.

References `cache_pool`, `log_info`, `memcached_flush_d`, `memcached_strerror_d`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, and `pool_release()`.

5.90.1.6 `stringer_t* cache_get (stringer_t * key)`

Retrieve data from memcached by key.

Parameters:

key a managed string containing a key to be passed to memcached.

Returns:

NULL on failure, or a pointer to a managed string containing the cached data on success.

Definition at line 145 of file cache.c.

References `cache_pool`, `data`, `length`, `log_info`, `memcached_get_d`, `memcached_strerror_d`, `mm_free()`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `st_char_get()`, `st_empty`, `st_import()`, `st_length_get()`, and `st_length_int()`.

Referenced by `register_session_get()`, `smtp_check_greylist()`, `smtp_fetch_autoreply()`, `stamp_counter_check()`, and `teacher_data_get()`.

5.90.1.7 `uint64_t cache_get_u64 (stringer_t * key)`

Retrieve a 64 bit value from memcached by key.

Parameters:

key a managed string containing a key to be passed to memcached.

Returns:

NULL on failure, or the 64 bit value cached data on success.

Definition at line 185 of file cache.c.

References cache_pool, data, length, log_info, memcached_get_d, memcached_strerror_d, mm_free(), PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by contact_abuse_checks(), register_abuse_checks(), and smtp_reply().

5.90.1.8 uint64_t cache_increment (stringer_t * key, uint64_t offset, uint64_t initial, time_t expiration)

Increment the value stored with a key by memcached by a specified offset.

See also:

memcached_increment_with_initial()

Note:

If the expiration value is MEMCACHED_EXPIRATION_NOT_ADD the key won't be added, but the initial value will be used as the return value.

Parameters:

key a managed string containing the name of the memcached key to be adjusted.

offset the offset by which the specified key's value should be incremented.

initial the initial value to seed the key if it does not already exist.

expiration the time, in seconds, when the cached key will expire.

Returns:

0 on failure, or the newly incremented value associated with the key, or the initial value, on success.

Definition at line 381 of file cache.c.

References cache_pool, log_pedantic, memcached_increment_with_initial_d, memcached_strerror_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by api_endpoint_auth(), contact_abuse_increment_history(), imap_login(), pop_pass(), portal_endpoint_auth(), register_abuse_increment_history(), serial_get(), serial_increment(), smtp_auth_login(), and smtp_auth_plain().

5.90.1.9 int_t cache_set (stringer_t * key, stringer_t * object, time_t expiration)

Set a value in memcached by key.

Parameters:

key a managed string containing a key to be passed to memcached.

object a managed string containing the new data to be associated with the supplied key.

expiration the expiration time of the cached data.

Returns:

0 on failure, or 1 on success.

Definition at line 231 of file cache.c.

References cache_pool, log_info, memcached_set_d, memcached_strerror_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by register_session_cache(), serial_reset(), smtp_check_greylist(), smtp_fetch_autoreply(), and teacher_data_save().

5.90.1.10 int_t cache_set_u64 (stringer_t * key, uint64_t value, time_t expiration)

Set a 64 bit value in memcached by key.

Parameters:

key a managed string containing a key to be passed to memcached.

value the new value to be associated with the supplied key.

expiration the expiration time of the cached data.

Returns:

NULL on failure, or the retrieved unsigned 64 bit cached data on success.

Definition at line 255 of file cache.c.

References cache_pool, log_info, memcached_set_d, memcached_strerror_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by smtp_reply().

5.90.1.11 int_t cache_silent_add (stringer_t * key, stringer_t * object, time_t expiration)

Add a new key/value pair to the memcached server, but suppress any error messages.

Parameters:

key a managed string with the name of the key to be added to the cache.

object a managed string containing the initial value of the new key.

expiration an expiration time, in seconds, for the newly cached data.

Definition at line 301 of file cache.c.

References cache_pool, memcached_add_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, and st_length_get().

Referenced by lock_get().

5.90.1.12 bool_t cache_start (void)

Definition at line 45 of file cache.c.

References magma_t::cache, cache_pool, magma_t::iface, log_critical, magma, MAGMA_CACHE_INSTANCES, memcached_behavior_set_d, memcached_create_d, memcached_free_d, memcached_server_add_with_weight_d, memcached_strerror_d, pool_alloc(), and pool_set_obj().

Referenced by process_start().

5.90.1.13 void cache_stop (void)

Destroy the memcached client pool and all active connections.

Returns:

This function returns no value.

Definition at line 104 of file cache.c.

References magma_t::cache, cache_pool, magma_t::iface, magma, memcached_free_d, pool_free(), and pool_get_obj().

Referenced by process_stop().

5.90.1.14 bool_t lib_load_cache (void)

Initialize the memcached library and bind dynamically to the exported functions that are required.

Returns:

true on success or false on failure.

Definition at line 16 of file cache.c.

References lib_symbols(), M_BIND, and symbol_t.

Referenced by lib_load().

5.90.1.15 const char* lib_version_cache (void)

Return the version string of the memcached library.

Returns:

a pointer to a character string containing the memcached library version information.

Definition at line 37 of file cache.c.

References memcached_lib_version_d.

Referenced by lib_load().

5.90.2 Variable Documentation**5.90.2.1 pool_t* cache_pool = NULL**

Definition at line 10 of file cache.c.

Referenced by cache_add(), cache_append(), cache_decrement(), cache_delete(), cache_flush(), cache_get(), cache_get_u64(), cache_increment(), cache_set(), cache_set_u64(), cache_silent_add(), cache_start(), and cache_stop().

5.91 src/providers/dime/signet-resolver/cache.c File Reference

```
#include <stdint.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <pthread.h>
#include "dime/signet-resolver/cache.h"
#include "dime/signet-resolver/dns.h"
#include "dime/signet-resolver/mrec.h"
#include "dime/signet-resolver/signet-ssl.h"
#include "dime/signet/keys.h"
#include "dime/signet/signet.h"
#include "dime/common/misc.h"
#include "dime/common/error.h"
```

Functions

- [cached_store_t * _get_cached_store_by_type](#) (cached_data_type_t dtype)
Get a pointer to a cached object store by its type.
- [char * _get_dime_dir_location](#) (const char *suffix)
Get the full pathname of the DIME base directory.
- [char * _get_cache_location](#) (void)
Get the full pathname of the DIME cache file.
- [int _set_cache_location](#) (const char *path)
Set the pathname of the DIME management record cache file.
- [cached_object_t * _create_cached_object](#) (cached_data_type_t dtype, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)
Allocate and initialize a new cached object.
- [void * _get_cache_obj_data](#) (cached_object_t *object)
Get the data associated with a cached object, freeing the parent cached object.
- [void _destroy_cache_entry](#) (cached_object_t *entry)
Destroy a cached object.
- [cached_object_t * _find_cached_object](#) (const char *oid, cached_store_t *store)
Find a cached object in a cached store.
- [cached_object_t * _find_cached_object_cmp](#) (const void *key, cached_store_t *store, cached_store_comparator_t cmpfn)
Find a cached object in a cached store using a custom comparator.
- [int _cached_object_exists](#) (const unsigned char *hashid, cached_store_t *store)
Check by a cached object's hashed id to see if it already exists in a cached object store.

- `int _cached_object_exists_cmp` (const void *key, `cached_store_t` *store, `cached_store_comparator_t` cmpfn)
Check to see if an object exists in a cached store by using a custom comparator function.
- `cached_object_t * _add_cached_object` (const char *id, `cached_store_t` *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)
Create and add a cached object to a cached store.
- `cached_object_t * _add_cached_object_forced` (const char *id, `cached_store_t` *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)
Create and add an object to the cache, forcing the removal of any existing object with a clashing id.
- `cached_object_t * _add_cached_object_cmp` (const char *id, const void *key, `cached_store_t` *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, `cached_store_comparator_t` cmpfn)
Create and add a cached object to a cached store using a custom comparator check for collisions.
- `cached_object_t * _add_cached_object_cmp_forced` (const char *id, const void *key, `cached_store_t` *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, `cached_store_comparator_t` cmpfn)
Create and add an object to the cache, forcing the removal of any existing object that matches the result of a custom comparison function.
- `int _remove_cached_object` (const char *oid, `cached_store_t` *store)
Remove an object from a cached store by name.
- `int _remove_cached_object_cmp` (const void *key, `cached_store_t` *store, `cached_store_comparator_t` cmpfn)
Remove an object from a cached store using a custom comparator function.
- `cached_object_t * _clone_cached_object` (const `cached_object_t` *obj)
Clone a deep copy of a cached object for user-safe retrieval.
- `void _dump_cache` (`cached_data_type_t` dtype, int do_data, int ephemeral)
Dump the content of the specified data cached store(s) to the console.
- `void _dump_cache_data` (FILE *fp, const `cached_object_t` *obj, int brief)
Dump information about a specified cached object to a file stream.
- `int _load_cache_contents` (void)
Persist the entire contents of the cache to local storage.
- `int _save_cache_contents` (void)
Persist the entire contents of the cache to local storage.
- `int _set_cache_permissions` (unsigned long flags)
Set the permissions flags for the object cache.
- `size_t _mem_append_serialized` (unsigned char **buf, size_t *blen, const unsigned char *data, size_t dlen)
Append a variable-length chunk of data to a dynamically allocated buffer for serialization.
- `size_t _mem_append_serialized_string` (unsigned char **buf, size_t *blen, const char *string)
Append a null-terminated string to a dynamically allocated buffer for serialization.
- `size_t _mem_append_serialized_array` (unsigned char **buf, size_t *blen, const unsigned char **array, size_t itemsize)
Append an array of fixed-size objects to a dynamically allocated buffer for serialization.

- `size_t _mem_append_serialized_str_array` (unsigned char **buf, size_t *blen, const char **array)
Append an array of null-terminated strings to a dynamically allocated buffer for serialization.
- `size_t _mem_append_serialized_array_cb` (unsigned char **buf, size_t *blen, const char **array, `custom_serializer_t` sfn)
Append an array of custom-formatted data to a dynamically allocated buffer for serialization.
- `unsigned int _deserialize_data` (unsigned char *dst, unsigned char **bufptr, const unsigned char *bufend, size_t len)
Deserialize a fixed-sized chunk of data from an input buffer.
- `unsigned char * _deserialize_vardata` (unsigned char **bufptr, const unsigned char *bufend, size_t *olen)
Deserialize a variable-length chunk of data from an input buffer.
- `unsigned int _deserialize_string` (char **dst, unsigned char **bufptr, const unsigned char *bufend)
Deserialize a null-terminated string from an input buffer.
- `unsigned int _deserialize_array` (unsigned char ***dst, unsigned char **bufptr, const unsigned char *bufend, size_t itemsize)
Deserialize an array of fixed-size objects from an input buffer.
- `unsigned int _deserialize_str_array` (char ***dst, unsigned char **bufptr, const unsigned char *bufend)
Deserialize an array of null-terminated string from an input buffer.
- `unsigned int _deserialize_array_cb` (unsigned char ***dst, unsigned char **bufptr, const unsigned char *bufend, `custom_deserializer_t` dfn)
Deserialize an array of custom-formatted data objects from an input buffer using a deserialization function.
- `int _is_object_expired` (`cached_object_t` *obj, int *refresh)
Check to see if a cached object needs to be evicted for exceeding its TTL or expiration.
- `cached_object_t * _unlink_object` (`cached_object_t` *object, int destroy, int stale)
Unlink a cached object from its doubly linked list.
- `cached_object_t * _replace_object` (`cached_object_t` *oobj, `cached_object_t` *nobj, int shadow)
Replace one object in the cache with another.
- `void _lock_cache_store` (`cached_store_t` *store)
Lock a cached store for thread-safe processing.
- `void _unlock_cache_store` (`cached_store_t` *store)
Unlock a cached store for use by other callers.
- `unsigned int _evict_if_stale` (`cached_object_t` **objptr)
Evict an item from the object cache if it is stale (has expired).
- `void * _deserialize_signet_cb` (void *data, size_t len)
Deserializes signet from a void pointer.
- `void * _serialize_signet_cb` (void *record, size_t *outlen)
Serializes a `signet_t` structure into a binary string.
- `void _dump_signet_cb` (FILE *fp, void *record, int brief)
Dumps a signet from a `signet_t` structure.
- `void _destroy_signet_cb` (void *record)
Destroys a `signet_t` structure.

Variables

- `cached_store_t cached_stores` [cached_data_signet+1]

5.91.1 Function Documentation

5.91.1.1 `cached_object_t* _add_cached_object (const char *id, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)`

Create and add a cached object to a cached store.

Parameters:

id a unique identifier to be associated with the cached object in the store.

store a pointer to the cached store that will hold the new cached object.

ttl an optional time-to-live value for the cached object, in seconds.

expiration an optional expiration date for the cached object, as a UTC time.

data a pointer to the object-specific data to be associated with the cache entry.

persists if set, the cached object will be persisted to disk when the cache is saved.

relaxed if set, use a relaxed cache policy; this ensures that even if the entry's TTL has expired, if its absolute (UTC) expiration has not been reached, it will not be evicted from the cache, but will delivery a refresh notification to the caller.

Returns:

a pointer to the newly allocated cached object for the specified data on success, or NULL on failure.

Definition at line 556 of file cache.c.

References `_clone_cached_object()`, `_compute_sha_hash()`, `_evict_if_stale()`, `_lock_cache_store()`, `_unlock_cache_store()`, `CACHE_PERM_ADD`, `cached_object::data`, `cached_store_t::dtype`, `cached_object::dtype`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_PERM`, `ERR_UNSPEC`, `cached_object::expiration`, `cached_store_t::head`, `cached_object::id`, `cached_object::next`, `cached_object::persists`, `cached_object::prev`, `PUSH_ERROR_SYSCALL`, `cached_object::relaxed`, `RET_ERROR_PTR`, `RET_ERROR_PTR_FMT`, `SHA_256_SIZE`, `cached_object::timestamp`, and `cached_object::ttl`.

Referenced by `_add_cached_object_forced()`, `_do_ocsp_validation()`, `_get_dime_record()`, and `_get_signet()`.

5.91.1.2 `cached_object_t* _add_cached_object_cmp (const char *id, const void *key, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, cached_store_comparator_t cmpfn)`

Create and add a cached object to a cached store using a custom comparator check for collisions.

Parameters:

id a unique identifier to be associated with the cached object in the store.

key a pointer to an object-specific key that will be passed to the custom comparison function.

store a pointer to the cached store that will hold the new cached object.

ttl an optional time-to-live value for the cached object, in seconds.

expiration an optional expiration date for the cached object, as a UTC time.

data a pointer to the object-specific data to be associated with the cache entry.

persists if set, the cached object will be persisted to disk when the cache is saved.

relaxed if set, use a relaxed cache policy; this ensures that even if the entry's TTL has expired, if its absolute (UTC) expiration has not been reached, it will not be evicted from the cache, but will delivery a refresh notification to the caller.

cmpfn a custom comparison function that will compare the key value to the objects in the specified cache dstore and determine whether the two objects match one another.

Returns:

a pointer to the newly allocated cached object for the specified data on success, or NULL on failure.

Definition at line 741 of file cache.c.

References `_clone_cached_object()`, `_compute_sha_hash()`, `_create_cached_object()`, `_evict_if_stale()`, `_lock_cache_store()`, `_unlock_cache_store()`, `CACHE_PERM_ADD`, `cached_object::data`, `cached_store_t::dtype`, `ERR_BAD_PARAM`, `ERR_PERM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::id`, `cached_object::next`, `cached_object::prev`, `RET_ERROR_PTR`, and `SHA_256_SIZE`.

Referenced by `_add_cached_object_cmp_forced()`, `_add_dnskey_entry_rsa()`, and `_add_ds_entry()`.

5.91.1.3 `cached_object_t* _add_cached_object_cmp_forced (const char *id, const void *key, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, cached_store_comparator_t cmpfn)`

Create and add an object to the cache, forcing the removal of any existing object that matches the result of a custom comparison function.

Parameters:

id a unique identifier to be associated with the cached object in the store.

key a pointer to an object-specific key that will be passed to the custom comparison function.

store a pointer to the cached store that will hold the new cached object.

ttl an optional time-to-live value for the cached object, in seconds.

expiration an optional expiration date for the cached object, as a UTC time.

data a pointer to the object-specific data to be associated with the cache entry.

persists if set, the cached object will be persisted to disk when the cache is saved.

relaxed if set, use a relaxed cache policy; this ensures that even if the entry's TTL has expired, if its absolute (UTC) expiration has not been reached, it will not be evicted from the cache, but will delivery a refresh notification to the caller.

cmpfn a custom comparison function that will compare the key value to the objects in the specified cache dstore and determine whether the two objects match one another.

Returns:

a pointer to the newly allocated cached object for the specified data on success, or NULL on failure.

Definition at line 835 of file cache.c.

References `_add_cached_object_cmp()`, `_clone_cached_object()`, `_create_cached_object()`, `_dbgprint()`, `_destroy_cache_entry()`, `_dump_cache_data()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_replace_object()`, `_verbose`, `CACHE_PERM_ADD`, `cached_object::data`, `cached_store_t::dtype`, `ERR_BAD_PARAM`, `ERR_PERM`, `ERR_UNSPEC`, and `RET_ERROR_PTR`.

Referenced by `_add_dnskey_entry_rsa()`.

5.91.1.4 `cached_object_t* _add_cached_object_forced (const char *id, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)`

Create and add an object to the cache, forcing the removal of any existing object with a clashing id.

Note:

If an object with the same identifier already exists, it will be stored as the "shadow" data of the newly created cached entry. This allows for shadowed data to be saved to disk on a cache save, even if the over-shadowing data is not marked as persistent.

Parameters:

id a unique identifier to be associated with the cached object in the store.

store a pointer to the cached store that will hold the new cached object.

tll an optional time-to-live value for the cached object, in seconds.

expiration an optional expiration date for the cached object, as a UTC time.

data a pointer to the object-specific data to be associated with the cache entry.

persists if set, the cached object will be persisted to disk when the cache is saved.

relaxed if set, use a relaxed cache policy; this ensures that even if the entry's TTL has expired, if its absolute (UTC) expiration has not been reached, it will not be evicted from the cache, but will delivery a refresh notification to the caller.

Returns:

a pointer to the newly allocated cached object for the specified data on success, or NULL on failure.

Definition at line 662 of file cache.c.

References `_add_cached_object()`, `_clone_cached_object()`, `_create_cached_object()`, `_dbgprint()`, `_destroy_cache_entry()`, `_dump_cache_data()`, `_find_cached_object()`, `_replace_object()`, `_verbose`, `CACHE_PERM_ADD`, `cached_object::data`, `cached_store_t::dtype`, `ERR_BAD_PARAM`, `ERR_PERM`, `ERR_UNSPEC`, and `RET_ERROR_PTR`.

Referenced by `_get_dime_record_from_file()`.

5.91.1.5 `int _cached_object_exists (const unsigned char * hashid, cached_store_t * store)`

Check by a cached object's hashed id to see if it already exists in a cached object store.

Parameters:

store a pointer to the cached store in which to search for the cached object.

hashid the (already) hashed unique id of the object to be located in the cache.

Returns:

1 if the object described was present in the cache, 0 if it was not, or -1 on error.

Definition at line 460 of file cache.c.

References `_evict_if_stale()`, `_lock_cache_store()`, `_unlock_cache_store()`, `CACHE_PERM_READ`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::id`, `cached_object::next`, `RET_ERROR_INT`, and `SHA_256_SIZE`.

Referenced by `_load_cache_contents()`.

5.91.1.6 `int _cached_object_exists_cmp (const void * key, cached_store_t * store, cached_store_comparator_t cmpfn)`

Check to see if an object exists in a cached store by using a custom comparator function.

Note:

This function is used when the simple unique object ID is not enough to identify an object.

Parameters:

key a pointer to an object-specific key that will be passed to the comparison function.

store a pointer to the cached store in which to search for the cached object.

cmpfn a comparison function that will compare the key value to the objects in the specified cached store, and determine whether the two objects match one another.

Returns:

1 if the object described was present in the cache, 0 if it was not, or -1 on error.

Definition at line 506 of file cache.c.

References `_evict_if_stale()`, `_lock_cache_store()`, `_unlock_cache_store()`, `CACHE_PERM_READ`, `cached_object::data`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::next`, and `RET_ERROR_INT`.

5.91.1.7 `cached_object_t* _clone_cached_object (const cached_object_t * obj)`

Clone a deep copy of a cached object for user-safe retrieval.

Definition at line 994 of file `cache.c`.

References `_get_cached_store_by_type()`, `cached_store_t::clone`, `cached_object::data`, `data`, `cached_store_t::deserialize`, `cached_object::dtype`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `cached_object::expiration`, `cached_object::id`, `cached_store_t::internal`, `cached_object::persists`, `PUSH_ERROR_SYSCALL`, `cached_object::relaxed`, `RET_ERROR_PTR`, `cached_store_t::serialize`, `store`, `cached_object::timestamp`, and `cached_object::ttl`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_find_cached_object()`, and `_find_cached_object_cmp()`.

5.91.1.8 `cached_object_t* _create_cached_object (cached_data_type_t dtype, unsigned long ttl, time_t expiration, void * data, int persists, int relaxed)`

Allocate and initialize a new cached object.

See also:

[`_add_cached_object\(\)`](#)

Note:

The data parameter specifies an object-specific value that will be passed to other parts of the cache management subsystem, such as functions for deletion, destruction, search, etc.

Parameters:

dtype the data type identifying the cached store to which this cached object will belong.

ttl if not 0, an optional time-to-live value in seconds specifying in how many seconds from now the cached object will reach expiration.

expiration if not 0, an optional UTC time value specifying the exact time at which this cached object will reach expiration.

data a pointer to the object-specific data to be associated with the cache entry.

persists if set, the cached object will be persisted to disk when the cache is saved.

relaxed if set, use a relaxed cache policy. A relaxed cache policy ensures that even if the entry's ttl has expired, if its absolute (UTC) expiration time has not been reached, it will not be evicted from the cache, but will allow for a notification to the caller that it is time for the cached entry to be refreshed.

Returns:

a pointer to a newly allocated cached object for the specified data on success, or NULL on failure.

Definition at line 244 of file `cache.c`.

References `cached_object::data`, `cached_object::dtype`, `ERR_NOMEM`, `ERR_UNSPEC`, `cached_object::expiration`, `cached_object::persists`, `PUSH_ERROR_SYSCALL`, `cached_object::relaxed`, `RET_ERROR_PTR`, `cached_object::timestamp`, and `cached_object::ttl`.

Referenced by `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, and `_get_dime_record()`.

5.91.1.9 `unsigned int _deserialize_array (unsigned char *** dst, unsigned char ** bufptr, const unsigned char * bufend, size_t itemsize)`

Deserialize an array of fixed-size objects from an input buffer.

Note:

This is the inverse function of [`_mem_append_serialized_array\(\)`](#).

Parameters:

- dst* a pointer to the address of an array of objects that will receive a copy of the deserialized data as an array of (NULL pointer terminated) fixed-size objects on success.
- bufptr* a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.
- bufend* a pointer to the end of the input buffer for validation purposes.
- itemsz* the size, in bytes, of the fixed-sized objects to be deserialized.

Returns:

1 on success or 0 on failure.

Definition at line 1876 of file cache.c.

References `_deserialize_data()`, `_ptr_chain_add()`, `_ptr_chain_free()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_UINT`.

Referenced by `_deserialize_dime_record_cb()`.

5.91.1.10 `unsigned int _deserialize_array_cb (unsigned char *** dst, unsigned char ** bufptr, const unsigned char * bufend, custom_deserializer_t dfn)`

Deserialize an array of custom-formatted data objects from an input buffer using a deserialization function.

Note:

This is the inverse function of [_mem_append_serialized_array_cb\(\)](#).

Parameters:

- dst* a pointer to the address of an array of data buffers that will receive a copy of the deserialized data as an array of (NULL pointer terminated) data buffers on success.
- bufptr* a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.
- bufend* a pointer to the end of the input buffer for validation purposes.
- dfn* a custom deserialization function that will be used to deserialize the user data from the input buffer.

Returns:

1 on success or 0 on failure.

Definition at line 2025 of file cache.c.

References `_deserialize_data()`, `_ptr_chain_add()`, `_ptr_chain_free()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, and `RET_ERROR_UINT`.

5.91.1.11 `unsigned int _deserialize_data (unsigned char * dst, unsigned char ** bufptr, const unsigned char * bufend, size_t len)`

Deserialize a fixed-sized chunk of data from an input buffer.

Note:

This is the inverse function of [_mem_append\(\)](#); *dst* must be at least *len* bytes long.

Parameters:

- dst* a pointer to the output buffer that will receive the deserialized data.
- bufptr* a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.

bufend a pointer to the end of the input buffer for validation purposes.

len the size of the data chunk to be deserialized from the input buffer.

Returns:

1 on success or 0 on failure.

Definition at line 1755 of file cache.c.

References ERR_BAD_PARAM, ERR_UNSPEC, RET_ERROR_UINT, and RET_ERROR_UINT_FMT.

Referenced by _deserialize_array(), _deserialize_array_cb(), _deserialize_dime_record_cb(), _deserialize_dnskey_record_cb(), _deserialize_ds_record_cb(), _deserialize_str_array(), and _deserialize_vardata().

5.91.1.12 void* _deserialize_signet_cb (void * data, size_t len)

Deserializes signet from a void pointer.

Parameters:

data Pointer to serial signet data.

len Length of serial signet data.

Returns:

Void pointer to a [signet_t](#) structure.

Definition at line 2321 of file cache.c.

References dime_sgnt_signet_binary_deserialize(), ERR_BAD_PARAM, ERR_UNSPEC, and RET_ERROR_PTR.

5.91.1.13 unsigned int _deserialize_str_array (char *** dst, unsigned char ** bufptr, const unsigned char * bufend)

Deserialize an array of null-terminated string from an input buffer.

Note:

This is the inverse function of [_mem_append_serialized_str_array\(\)](#).

Parameters:

dst a pointer to the address of an array of strings that will receive a copy of the deserialized data as an array of (NULL pointer terminated) null-terminated strings on success.

bufptr a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.

bufend a pointer to the end of the input buffer for validation purposes.

Returns:

1 on success or 0 on failure.

Definition at line 1955 of file cache.c.

References _deserialize_data(), _deserialize_string(), _ptr_chain_add(), _ptr_chain_free(), ERR_BAD_PARAM, ERR_UNSPEC, and RET_ERROR_UINT.

Referenced by _deserialize_dime_record_cb().

5.91.1.14 unsigned int _deserialize_string (char ** *dst*, unsigned char ** *bufptr*, const unsigned char * *bufend*)

Deserialize a null-terminated string from an input buffer.

Note:

This is the inverse function of [_mem_append_serialized_string\(\)](#).

Parameters:

dst a pointer to the address of a string that will receive a copy of the deserialized data as a null-terminated string on success.

bufptr a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.

bufend a pointer to the end of the input buffer for validation purposes.

Returns:

1 on success or 0 on failure.

Definition at line 1826 of file cache.c.

References `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_UINT`.

Referenced by `_deserialize_dime_record_cb()`, `_deserialize_dnskey_record_cb()`, `_deserialize_ds_record_cb()`, and `_deserialize_str_array()`.

5.91.1.15 unsigned char* _deserialize_vardata (unsigned char ** *bufptr*, const unsigned char * *bufend*, size_t * *olen*)

Deserialize a variable-length chunk of data from an input buffer.

Note:

This is the inverse function of `_mem_append_serialized`.

Parameters:

bufptr a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.

bufend a pointer to the end of the input buffer for validation purposes.

olen an optional pointer to receive the size of the deserialized data on success.

Returns:

a pointer to a newly allocated buffer containing the deserialized data on success, or NULL on failure.

Definition at line 1784 of file cache.c.

References `_deserialize_data()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, and `RET_ERROR_UINT`.

Referenced by `_deserialize_dnskey_record_cb()`, and `_deserialize_ds_record_cb()`.

5.91.1.16 void _destroy_cache_entry (cached_object_t * *entry*)

Destroy a cached object.

Parameters:

entry a pointer to the cached object to be destroyed.

Definition at line 316 of file cache.c.

References `_clear_error_stack()`, `_get_cached_store_by_type()`, `cached_object::data`, `cached_store_t::destructor`, `cached_object::dtype`, `dump_error_stack()`, `get_last_error()`, and `store`.

Referenced by `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_get_cache_obj_data()`, `_get_dime_record()`, `_load_cache_contents()`, and `_replace_object()`.

5.91.1.17 `void _destroy_signet_cb (void * record)`

Destroys a [signet_t](#) structure.

Parameters:

record Void pointer to a [signet_t](#) structure to be destroyed.

Definition at line 2399 of file cache.c.

References `dime_sgnt_signet_destroy()`.

5.91.1.18 `void _dump_cache (cached_data_type_t dtype, int do_data, int ephemeral)`

Dump the content of the specified data cached store(s) to the console.

Parameters:

dtype the type of the object cached store to be dumped, or `cached_data_unknown` for all stores.

do_data if non-zero, dump the type-specific data associated with the cached object.

ephemeral if non-zero, print out ephemeral as well as persistent cache data.

Definition at line 1068 of file cache.c.

References `_dump_cache_data()`, `_get_chr_date()`, `_lock_cache_store()`, `_unlock_cache_store()`, `cached_data_unknown`, `cached_object::data`, `cached_store_t::dtype`, `cached_object::expiration`, `cached_object::next`, `cached_object::persists`, `cached_object::relaxed`, `cached_object::shadow`, `cached_object::timestamp`, and `cached_object::ttl`.

5.91.1.19 `void _dump_cache_data (FILE * fp, const cached_object_t * obj, int brief)`

Dump information about a specified cached object to a file stream.

Parameters:

fp a pointer to the file stream across which the data should be dumped.

obj a pointer to the cached object to have its details dumped.

brief if true, print a brief one-line description of the object; otherwise, dump full object information.

Definition at line 1163 of file cache.c.

References `_clear_error_stack()`, `_get_cached_store_by_type()`, `cached_object::data`, `cached_object::dtype`, `cached_store_t::dump`, `dump_error_stack()`, `get_last_error()`, and `store`.

Referenced by `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_dump_cache()`, `_load_cache_contents()`, and `_unlink_object()`.

5.91.1.20 `void _dump_signet_cb (FILE * fp, void * record, int brief)`

Dumps a signet from a [signet_t](#) structure.

Parameters:

fp File descriptor that specifies the output destination.
record Void pointer to a [signet_t](#) structure.
brief TODO

Definition at line 2377 of file cache.c.

References `dime_sgnt_signet_dump()`.

5.91.1.21 unsigned int _evict_if_stale (cached_object_t ** objptr)

Evict an item from the object cache if it is stale (has expired).

Returns:

1 if the object was evicted for being stale or 0 if it was not.

Definition at line 2287 of file cache.c.

References `_clear_error_stack()`, `_is_object_expired()`, `_unlink_object()`, `dump_error_stack()`, and `get_last_error()`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_remove_cached_object()`, and `_remove_cached_object_cmp()`.

5.91.1.22 cached_object_t* _find_cached_object (const char * oid, cached_store_t * store)

Find a cached object in a cached store.

Parameters:

oid a null-terminated string containing the unique name or identifier of the cached object.
store a pointer to the cached store to be searched for the target object.

Returns:

a pointer to the specified cached object, if found, or NULL on failure.

Definition at line 355 of file cache.c.

References `_clone_cached_object()`, `_compute_sha_hash()`, `_evict_if_stale()`, `_lock_cache_store()`, `_unlock_cache_store()`, `CACHE_PERM_READ`, `ERR_BAD_PARAM`, `ERR_PERM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::id`, `cached_object::next`, `RET_ERROR_PTR`, and `SHA_256_SIZE`.

Referenced by `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_do_ocsp_validation()`, `_get_dime_record()`, and `_get_signet()`.

5.91.1.23 cached_object_t* _find_cached_object_cmp (const void * key, cached_store_t * store, cached_store_comparator_t cmpfn)

Find a cached object in a cached store using a custom comparator.

Note:

This function is used when the simple unique object ID is not sufficient to identify an object.

Parameters:

key a pointer to an object-specific key that will be passed to the custom comparison function.

store a pointer to the cached store in which to search for the cached object.

cmpfn a custom comparison function that will compare the key value to the objects in the specified cached store and determine whether the two objects match one another.

Returns:

a pointer to the found cached object on success, or NULL on failure.

Definition at line 412 of file cache.c.

References `_clone_cached_object()`, `_evict_if_stale()`, `_lock_cache_store()`, `_unlock_cache_store()`, `CACHE_PERM_READ`, `cached_object::data`, `ERR_BAD_PARAM`, `ERR_PERM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::next`, and `RET_ERROR_PTR`.

Referenced by `_add_cached_object_cmp_forced()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, and `_load_dnskey_file()`.

5.91.1.24 `char* _get_cache_location (void)`

Get the full pathname of the DIME cache file.

Note:

The default location returned by this function can be overridden by setting the `DIME_CACHE_FILE` environment variable.

Returns:

NULL on failure, or a null-terminated string containing the filename of the DIME object cache file on success. {free}

Definition at line 154 of file cache.c.

References `_dbgprint()`, `_get_dime_dir_location()`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_load_cache_contents()`, and `_save_cache_contents()`.

5.91.1.25 `void* _get_cache_obj_data (cached_object_t * object)`

Get the data associated with a cached object, freeing the parent cached object.

Note:

This function is useful as a wrapper for returning cached data without returning the meta-information associated with the same data that is stored inside its `cached_object_t` holder. If the object belongs to an internal store, no memory will be freed.

Parameters:

object a pointer to the cached object to have its opaque data returned, and enclosing structure freed.

Returns:

NULL on failure, or a pointer to the cached object's data on success.

Definition at line 281 of file cache.c.

References `_destroy_cache_entry()`, `_get_cached_store_by_type()`, `cached_object::data`, `cached_object::dtype`, `ERR_UNSPEC`, `cached_store_t::internal`, `RET_ERROR_PTR`, and `store`.

Referenced by `_add_dnskey_entry_rsa()`, `_add_ds_entry()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, and `_get_signet()`.

5.91.1.26 `cached_store_t* _get_cached_store_by_type (cached_data_type_t dtype)`

Get a pointer to a cached object store by its type.

Parameters:

dtype the numerical identifier of the cached data type.

Returns:

a pointer to the requested cached object store on success, or NULL on failure.

Definition at line 46 of file cache.c.

Referenced by `_clone_cached_object()`, `_destroy_cache_entry()`, `_dump_cache_data()`, `_get_cache_obj_data()`, `_load_cache_contents()`, `_replace_object()`, and `_unlink_object()`.

5.91.1.27 `char* _get_dime_dir_location (const char * suffix)`

Get the full pathname of the DIME base directory.

Parameters:

suffix if not NULL, an optional suffix containing a filename or path to be appended to the end of the DIME directory name.

Returns:

NULL on failure, or a null-terminated string containing the requested DIME directory pathname on success.

Definition at line 61 of file cache.c.

References `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, and `str_printf()`.

Referenced by `_get_cache_location()`, `_initialize_resolver()`, and `_ssl_initialize()`.

5.91.1.28 `int _is_object_expired (cached_object_t * obj, int * refresh)`

Check to see if a cached object needs to be evicted for exceeding its TTL or expiration.

Note:

An object has "expired" and needs to be evicted from the cache if it has a non-zero TTL from its cache date that has been reached, or a non-zero absolute expiration date that has been exceeded. If the object's cache policy is relaxed, a TTL expiration advises a "refresh" if no absolute expiration has occurred.

Parameters:

obj a pointer to the cached object to have its age examined.

refresh an optional pointer to a value that, if the cached object has a relaxed expiration policy, will be set to 1 if it needs to be refreshed or 0 if it does not.

Returns:

1 if the object has expired or 0 if it has not; -1 if a general error has occurred.

Definition at line 2093 of file cache.c.

References `ERR_BAD_PARAM`, `ERR_UNSPEC`, `cached_object::expiration`, `PUSH_ERROR_SYSCALL`, `cached_object::relaxed`, `RET_ERROR_INT`, `cached_object::timestamp`, and `cached_object::ttl`.

Referenced by `_evict_if_stale()`, and `_get_dime_record()`.

5.91.1.29 int _load_cache_contents (void)

Persist the entire contents of the cache to local storage.

Returns:

1 if the cache was loaded successfully, 0 if it was created, or -1 on general failure.

Definition at line 1196 of file cache.c.

References `_cached_object_exists()`, `_clear_error_stack()`, `_dbgprint()`, `_destroy_cache_entry()`, `_dump_cache_data()`, `_get_cache_location()`, `_get_cached_store_by_type()`, `_get_chr_date()`, `_lock_cache_store()`, `_unlock_cache_store()`, `_verbose`, `CACHE_PERM_LOAD`, `cached_object::data`, `cached_store_t::deserialize`, `cached_object::dtype`, `dump_error_stack()`, `ERR_NOMEM`, `ERR_PERM`, `ERR_UNSPEC`, `cached_object::expiration`, `cached_store_t::head`, `cached_object::id`, `cached_object::next`, `cached_object::persists`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, `store`, `cached_object::timestamp`, and `cached_object::ttl`.

5.91.1.30 void _lock_cache_store (cached_store_t * store)

Lock a cached store for thread-safe processing.

Parameters:

store a pointer to the cached store to be locked.

Definition at line 2261 of file cache.c.

References `cached_store_t::lock`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_dump_cache()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_fixup_dnskey_validation()`, `_load_cache_contents()`, `_remove_cached_object()`, `_remove_cached_object_cmp()`, and `_save_cache_contents()`.

5.91.1.31 size_t _mem_append_serialized (unsigned char ** buf, size_t * blen, const unsigned char * data, size_t dlen)

Append a variable-length chunk of data to a dynamically allocated buffer for serialization.

Parameters:

buf a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.

blen a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.

data a pointer to a buffer holding the data that will be appended to the end of the output buffer.

dlen the size, in bytes, of the data buffer to be appended to the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1524 of file cache.c.

References `_mem_append()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

Referenced by `_mem_append_serialized_array()`, `_mem_append_serialized_array_cb()`, `_mem_append_serialized_str_array()`, `_serialize_dnskey_record_cb()`, and `_serialize_ds_record_cb()`.

5.91.1.32 size_t _mem_append_serialized_array (unsigned char ** buf, size_t * blen, const unsigned char ** array, size_t itemsz)

Append an array of fixed-size objects to a dynamically allocated buffer for serialization.

Parameters:

buf a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.
blen a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.
array an array of pointers to fixed-size objects (ending with a NULL pointer) to be appended to the end of the output buffer.
*itemsiz*e the size, in bytes, of the array elements to be deserialized.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1585 of file cache.c.

References `_mem_append()`, `_mem_append_serialized()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

Referenced by `_serialize_dime_record_cb()`.

5.91.1.33 `size_t _mem_append_serialized_array_cb (unsigned char ** buf, size_t * blen, const char ** array, custom_serializer_t sfn)`

Append an array of custom-formatted data to a dynamically allocated buffer for serialization.

Note:

This function uses a custom serialization function in order to output the user data properly.

Parameters:

buf a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.
blen a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.
array an array of pointers to custom-formatted data buffers (ending with a NULL pointer) to be appended to the end of the output buffer using a custom serialization function.
sfn a custom serialization function that will be used to serialize the user data into the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1698 of file cache.c.

References `_mem_append()`, `_mem_append_serialized()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

5.91.1.34 `size_t _mem_append_serialized_str_array (unsigned char ** buf, size_t * blen, const char ** array)`

Append an array of null-terminated strings to a dynamically allocated buffer for serialization.

Parameters:

buf a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.
blen a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.
array an array of pointers to null-terminated strings (ending with a NULL pointer) to be appended to the end of the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1640 of file cache.c.

References `_mem_append()`, `_mem_append_serialized()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

Referenced by `_serialize_dime_record_cb()`.

5.91.1.35 `size_t _mem_append_serialized_string (unsigned char ** buf, size_t * blen, const char * string)`

Appended a null-terminated string to a dynamically allocated buffer for serialization.

Parameters:

- buf* a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.
- blen* a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.
- string* a null-terminated string that will be appended to the end of the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1552 of file cache.c.

References `_mem_append()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

Referenced by `_serialize_dime_record_cb()`, `_serialize_dnskey_record_cb()`, and `_serialize_ds_record_cb()`.

5.91.1.36 `int _remove_cached_object (const char * oid, cached_store_t * store)`

Remove an object from a cached store by name.

Note:

The caller is still responsible for freeing the cached object and its underlying data.

Parameters:

- oid* the unique identifier of the cached object to be removed.
- store* a pointer to the cached store from which the specified object will be removed.

Returns:

1 if the object was successfully removed or 0 if it couldn't be found; -1 on general error.

Definition at line 903 of file cache.c.

References `_compute_sha_hash()`, `_evict_if_stale()`, `_lock_cache_store()`, `_unlink_object()`, `_unlock_cache_store()`, `CACHE_PERM_DELETE`, `ERR_BAD_PARAM`, `ERR_PERM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::id`, `cached_object::next`, `RET_ERROR_INT`, and `SHA_256_SIZE`.

5.91.1.37 `int _remove_cached_object_cmp (const void * key, cached_store_t * store, cached_store_comparator_t cmpfn)`

Remove an object from a cached store using a custom comparator function.

Note:

The caller is still responsible for freeing the cached object and its underlying data.

Parameters:

- key* a pointer to an object-specific key that will be passed to the custom comparison function.
- store* a pointer to the cached store from which the specified object will be removed.
- cmpfn* a custom comparison function that will compare the key value to the objects in the specified cached store and determine whether the two objects match one another.

Returns:

1 if the object was successfully removed or 0 if it couldn't be found; -1 on general error.

Definition at line 953 of file cache.c.

References `_evict_if_stale()`, `_lock_cache_store()`, `_unlink_object()`, `_unlock_cache_store()`, `CACHE_PERM_DELETE`, `cached_object::data`, `ERR_BAD_PARAM`, `ERR_PERM`, `cached_store_t::head`, `cached_object::next`, and `RET_ERROR_INT`.

5.91.1.38 `cached_object_t* _replace_object (cached_object_t * oobj, cached_object_t * nobj, int shadow)`

Replace one object in the cache with another.

Note:

If shadow is not set, then the old cached object will be automatically destroyed. Otherwise this function also makes the old cached object the new cached object's "shadow" data.

Parameters:

oobj a pointer to the old cached object to be replaced in its cached store with the new object.

nobj a pointer to the new cached object that will replace the old object in the cached store.

shadow if set, preserve the old object as the "shadow" value of the new object.

Returns:

NULL on failure, or a pointer to the new cached object on success.

Definition at line 2208 of file cache.c.

References `_destroy_cache_entry()`, `_get_cached_store_by_type()`, `cached_object::dtype`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::id`, `cached_object::next`, `cached_object::prev`, `RET_ERROR_PTR`, `SHA_256_SIZE`, `cached_object::shadow`, and `store`.

Referenced by `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, and `_get_dime_record()`.

5.91.1.39 `int _save_cache_contents (void)`

Persist the entire contents of the cache to local storage.

Returns:

0 on success or -1 on failure.

Definition at line 1414 of file cache.c.

References `_clear_error_stack()`, `_dbgprint()`, `_get_cache_location()`, `_lock_cache_store()`, `_unlock_cache_store()`, `CACHE_PERM_LOAD`, `cached_object::data`, `dump_error_stack()`, `ERR_PERM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::next`, `cached_object::persists`, `PUSH_ERROR_FMT`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, and `cached_object::shadow`.

Referenced by `_do_ocsp_validation()`, and `_get_signet()`.

5.91.1.40 `void* _serialize_signet_cb (void * record, size_t * outlen)`

Serializes a [signet_t](#) structure into a binary string.

Parameters:

record Void pointer to a [signet_t](#) structure to be serialized.

outlen Pointer to the length of the returned string.

Returns:

Pointer to a serialized signet.

Definition at line 2343 of file cache.c.

References dime_sgnt_signet_binary_serialize(), ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

5.91.1.41 int _set_cache_location (const char * *path*)

Set the pathname of the DIME management record cache file.

Parameters:

path a null-terminated string containing the fully qualified path of the cache file to be used.

Returns:

0 on success or -1 on failure.

Definition at line 199 of file cache.c.

References _dbgprint(), ERR_BAD_PARAM, ERR_NOMEM, PUSH_ERROR_SYSCALL, and RET_ERROR_INT.

5.91.1.42 int _set_cache_permissions (unsigned long *flags*)

Set the permissions flags for the object cache.

Parameters:

flags

Returns:

-1 on error or 0 if the permissions were set successfully.

Definition at line 1503 of file cache.c.

References CACHE_PERM_ALL_FLAGS, ERR_BAD_PARAM, and RET_ERROR_INT.

5.91.1.43 cached_object_t* _unlink_object (cached_object_t * *object*, int *destroy*, int *stale*)

Unlink a cached object from its doubly linked list.

Parameters:

object a pointer to the cached object to be delinked.

destroy if set, deallocate the specified cached object after unlinking.

stale if set, the cache removal was performed because of a stale entry.

Returns:

a pointer to the next cached object in the store, or NULL if at the end of the store.

Definition at line 2140 of file cache.c.

References `_clear_error_stack()`, `_dbgprint()`, `_dump_cache_data()`, `_get_cached_store_by_type()`, `_verbose`, `cached_object::data`, `cached_store_t::destructor`, `cached_object::dtype`, `cached_store_t::dump`, `dump_error_stack()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `get_last_error()`, `cached_store_t::head`, `cached_object::next`, `cached_object::prev`, `RET_ERROR_PTR`, and `store`.

Referenced by `_do_ocsp_validation()`, `_evict_if_stale()`, `_remove_cached_object()`, and `_remove_cached_object_cmp()`.

5.91.1.44 void _unlock_cache_store (cached_store_t * store)

Unlock a cached store for use by other callers.

Parameters:

store a pointer to the cached store to be unlocked.

Definition at line 2274 of file cache.c.

References `cached_store_t::lock`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_dump_cache()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_fixup_dnskey_validation()`, `_load_cache_contents()`, `_remove_cached_object()`, `_remove_cached_object_cmp()`, and `_save_cache_contents()`.

5.91.2 Variable Documentation

5.91.2.1 cached_store_t cached_stores[cached_data_signet+1]

Initial value:

```
{
    { cached_data_unknown, "unknown", 0, NULL, PTHREAD_MUTEX_INITIALIZER, NULL, NULL, NULL, NULL, NULL },
    { cached_data_drec, "DIME management records", 0, NULL, PTHREAD_MUTEX_INITIALIZER, &_destroy_dime_record_cb, &_serialize_dime_record_cb, &_deserialize_dime_record_cb, &_dump_dime_record_cb, NULL },
    { cached_data_dnskey, "DNSKEY records", 1, NULL, PTHREAD_MUTEX_INITIALIZER, &_destroy_dnskey_record_cb, &_serialize_dnskey_record_cb, &_deserialize_dnskey_record_cb, &_dump_dnskey_record_cb, &_clone_dnskey_record_cb },
    { cached_data_ds, "DS records", 1, NULL, PTHREAD_MUTEX_INITIALIZER, &_destroy_ds_record_cb, &_serialize_ds_record_cb, &_deserialize_ds_record_cb, &_dump_ds_record_cb, NULL },
    { cached_data_ocsp, "OCSP", 1, NULL, PTHREAD_MUTEX_INITIALIZER, &_destroy_ocsp_response_cb, &_serialize_ocsp_response_cb, &_deserialize_ocsp_response_cb, &_dump_ocsp_response_cb, NULL },
    { cached_data_signet, "signets", 0, NULL, PTHREAD_MUTEX_INITIALIZER, &_destroy_signet_cb, &_serialize_signet_cb, &_deserialize_signet_cb, &_dump_signet_cb, NULL }
}
```

Definition at line 26 of file cache.c.

Referenced by `_add_dnskey_entry_rsa()`, `_add_ds_entry()`, `_do_ocsp_validation()`, `_fixup_dnskey_validation()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, `_get_signet()`, and `_load_dnskey_file()`.

5.92 src/engine/config/cache/cache.h File Reference

Data Structures

- struct [cache_keys_t](#)
- struct [cache_t](#)

Functions

- void [cache_free](#) (void)
Free magma's cache server configuration options.
- [bool_t](#) [cache_validate](#) (void)
Validate all of the configured magma cache server instances.
- void [cache_output_settings](#) (void)
Log the full contents of the magma cache configuration settings.
- [cache_t](#) * [cache_alloc](#) (uint32_t [number](#))
Allocate a new cache server configuration entry and initialize it to its default values.
- [bool_t](#) [cache_config](#) ([stringer_t](#) *name, [stringer_t](#) *value)
Set the value of a cache server configuration entry by name.
- [bool_t](#) [cache_set_value](#) ([cache_keys_t](#) *setting, [cache_t](#) *cache, [stringer_t](#) *value)
Set the value of a key for a specified cache server configuration entry.
- void [cache_output_help](#) (void)
Log all cache key information to be returned via "magma -h".

5.92.1 Function Documentation

5.92.1.1 [cache_t](#)* [cache_alloc](#) (uint32_t *number*)

Allocate a new cache server configuration entry and initialize it to its default values.

Parameters:

number the index of the cache server instance in the global cache server array.

Returns:

NULL on failure, or a pointer to the requested cache server configuration entry on success.

Definition at line 64 of file `cache.c`.

References `magma_t::cache`, `cache_keys`, `cache_set_value()`, `magma_t::iface`, `log_critical`, `magma`, and `mm_alloc()`.

Referenced by `cache_config()`.

5.92.1.2 bool_t cache_config (stringer_t * name, stringer_t * value)

Set the value of a cache server configuration entry by name.

Note:

This function is designed to operate on lines from a configuration source, like a file or database. When a named server is referenced for the first time, a new cache server configuration instance will be allocated.

Parameters:

name a managed string containing the human-readable cache server configuration parameter to be set.

value a managed string containing the new value of the cache server config key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 425 of file cache.c.

References bracket_extract_pl(), cache_alloc(), cache_keys, cache_set_value(), CONSTANT, log_critical, MAGMA_CACHE_INSTANCES, NULLER, pl_char_get(), pl_empty(), pl_length_get(), PLACER, placer_t, st_char_get(), st_cmp_ci_eq(), st_cmp_ci_starts(), st_length_get(), st_length_int(), and uint32_conv_bl().

Referenced by config_load_cmdline_settings(), config_load_database_settings(), and config_load_file_settings().

5.92.1.3 void cache_free (void)

Free magma's cache server configuration options.

Note:

This function assumes all worker threads have finished, and should only be called during the normal shutdown process.

Parameters:

This function returns no value.

Definition at line 17 of file cache.c.

References magma_t::cache, cache_keys, magma_t::iface, log_pedantic, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_ENUM, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_CACHE_INSTANCES, mm_free(), cache_keys_t::norm, ns_empty(), ns_free(), cache_keys_t::offset, st_empty, st_free(), type(), and multi_t::type.

Referenced by config_free().

5.92.1.4 void cache_output_help (void)

Log all cache key information to be returned via "magma -h".

Returns:

This function returns no value.

Definition at line 484 of file cache.c.

References multi_t::bl, cache_keys, config_output_value_generic(), log_info, log_options, M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE_DISABLE, M_LOG_TIME_DISABLE, cache_keys_t::norm, cache_keys_t::required, and multi_t::val.

Referenced by config_output_help().

5.92.1.5 void cache_output_settings (void)

Log the full contents of the magma cache configuration settings.

Note:

This logs all the details of each memcached instance configured to work with magma.

Returns:

This function returns no value.

Definition at line 207 of file cache.c.

References magma_t::cache, cache_keys, magma_t::iface, log_info, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_CACHE_INSTANCES, cache_keys_t::norm, ns_empty(), cache_keys_t::offset, st_char_get(), st_empty, st_length_int(), multi_t::type, and type().

Referenced by config_output_settings().

5.92.1.6 bool_t cache_set_value (cache_keys_t * setting, cache_t * cache, stringer_t * value)

Set the value of a key for a specified cache server configuration entry.

Note:

This function will allocate space for a copy of the key value, and return an error if it is not able to convert it to the proper key data type.

Parameters:

setting a pointer to the cache server key to be set.

cache a pointer to the cache server configuration entry to be modified.

value a managed string containing the new value of the key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 282 of file cache.c.

References multi_t::binary, CONSTANT, CONTIGUOUS, HEAP, multi_t::i32, multi_t::i64, multi_t::i8, int32_conv_st(), int64_conv_st(), int8_conv_st(), log_critical, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, MANAGED_T, cache_keys_t::name, cache_keys_t::norm, multi_t::ns, ns_dupe(), ns_empty(), ns_free(), ns_import(), cache_keys_t::offset, multi_t::st, st_char_get(), st_cmp_ci_eq(), st_dupe_opts(), st_empty, st_free(), st_length_get(), type(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, uint16_conv_st(), uint32_conv_st(), uint64_conv_st(), uint8_conv_st(), and multi_t::val.

Referenced by cache_alloc(), and cache_config().

5.92.1.7 bool_t cache_validate (void)

Validate all of the configured magma cache server instances.

Note:

This makes set that all required config keys have been set, and that at least one host has been configured.

Returns:

true if all cache servers have been validated, or false on failure.

Definition at line 93 of file cache.c.

References magma_t::cache, cache_keys, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, magma_t::iface, log_critical, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_CACHE_INSTANCES, cache_keys_t::norm, ns_empty(), cache_keys_t::offset, st_empty, multi_t::type, type(), multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by config_validate_settings().

5.93 src/providers/dime/signet-resolver/cache.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/types.h>
#include <pwd.h>
#include "dime/common/error.h"
```

Data Structures

- struct [cached_object](#)
- struct [cached_store_t](#)

Defines

- #define [CACHE_PERM_LOAD](#) 0x1
- #define [CACHE_PERM_SAVE](#) 0x2
- #define [CACHE_PERM_READ](#) 0x4
- #define [CACHE_PERM_ADD](#) 0x8
- #define [CACHE_PERM_DELETE](#) 0x16
- #define [CACHE_PERM_NONE](#) 0
- #define [CACHE_PERM_ALL_FLAGS](#) (CACHE_PERM_LOAD | CACHE_PERM_SAVE | CACHE_PERM_READ | CACHE_PERM_ADD | CACHE_PERM_DELETE)
- #define [CACHE_PERM_DEFAULT](#) CACHE_PERM_ALL_FLAGS

Typedefs

- typedef struct [cached_object](#) [cached_object_t](#)
- typedef int(* [cached_store_comparator_t](#))(const void *, const void *)
- typedef unsigned int(* [custom_deserializer_t](#))(unsigned char **, unsigned char **, const unsigned char *)
- typedef size_t(* [custom_serializer_t](#))(unsigned char **, size_t *, const void *)
- typedef void(* [custom_dumper_t](#))(FILE *, void *, int)

Enumerations

- enum [cached_data_type_t](#) {
[cached_data_unknown](#) = 0, [cached_data_drec](#) = 1, [cached_data_dnskey](#) = 2, [cached_data_ds](#) = 3,
[cached_data_ocsp](#) = 4, [cached_data_signet](#) = 5 }

Functions

- [PUBLIC_FUNC_DECL](#) (int, load_cache_contents, void)
- [PUBLIC_FUNC_DECL](#) (int, save_cache_contents, void)
- [PUBLIC_FUNC_DECL](#) (char *, get_dime_dir_location, const char *suffix)
- [PUBLIC_FUNC_DECL](#) (char *, get_cache_location, void)

- [PUBLIC_FUNC_DECL](#) (int, set_cache_location, const char *path)
- [PUBLIC_FUNC_DECL](#) (int, set_cache_permissions, unsigned long flags)
- [PUBLIC_FUNC_DECL](#) ([cached_object_t](#) *, find_cached_object, const char *oid, [cached_store_t](#) *store)
- [PUBLIC_FUNC_DECL](#) ([cached_object_t](#) *, find_cached_object_cmp, const void *key, [cached_store_t](#) *store, [cached_store_comparator_t](#) cmpfn)
- [PUBLIC_FUNC_DECL](#) (int, cached_object_exists, const unsigned char *hashid, [cached_store_t](#) *store)
- [PUBLIC_FUNC_DECL](#) (int, cached_object_exists_cmp, const void *key, [cached_store_t](#) *store, [cached_store_comparator_t](#) cmpfn)
- [PUBLIC_FUNC_DECL](#) ([cached_object_t](#) *, add_cached_object, const char *id, [cached_store_t](#) *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)
- [PUBLIC_FUNC_DECL](#) ([cached_object_t](#) *, add_cached_object_cmp, const char *id, const void *key, [cached_store_t](#) *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, [cached_store_comparator_t](#) cmpfn)
- [PUBLIC_FUNC_DECL](#) ([cached_object_t](#) *, add_cached_object_forced, const char *id, [cached_store_t](#) *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)
- [PUBLIC_FUNC_DECL](#) ([cached_object_t](#) *, add_cached_object_cmp_forced, const char *id, const void *key, [cached_store_t](#) *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, [cached_store_comparator_t](#) cmpfn)
- [PUBLIC_FUNC_DECL](#) (int, remove_cached_object, const char *oid, [cached_store_t](#) *store)
- [PUBLIC_FUNC_DECL](#) (int, remove_cached_object_cmp, const void *key, [cached_store_t](#) *store, [cached_store_comparator_t](#) cmpfn)
- [PUBLIC_FUNC_DECL](#) (void, destroy_cache_entry, [cached_object_t](#) *entry)
- [PUBLIC_FUNC_DECL](#) (void *, get_cache_obj_data, [cached_object_t](#) *object)
- [cached_object_t](#) * [_create_cached_object](#) ([cached_data_type_t](#) dtype, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)

Allocate and initialize a new cached object.

- [cached_store_t](#) * [_get_cached_store_by_type](#) ([cached_data_type_t](#) dtype)

Get a pointer to a cached object store by its type.

- void [_dump_cache](#) ([cached_data_type_t](#) dtype, int do_data, int ephemeral)

Dump the content of the specified data cached store(s) to the console.

- void [_dump_cache_data](#) (FILE *fp, const [cached_object_t](#) *obj, int brief)

Dump information about a specified cached object to a file stream.

- [cached_object_t](#) * [_clone_cached_object](#) (const [cached_object_t](#) *obj)

Clone a deep copy of a cached object for user-safe retrieval.

- size_t [_mem_append_serialized](#) (unsigned char **buf, size_t *blen, const unsigned char *data, size_t dlen)

Append a variable-length chunk of data to a dynamically allocated buffer for serialization.

- size_t [_mem_append_serialized_string](#) (unsigned char **buf, size_t *blen, const char *string)

Append a null-terminated string to a dynamically allocated buffer for serialization.

- size_t [_mem_append_serialized_array](#) (unsigned char **buf, size_t *blen, const unsigned char **array, size_t itemsz)

Append an array of fixed-size objects to a dynamically allocated buffer for serialization.

- size_t [_mem_append_serialized_str_array](#) (unsigned char **buf, size_t *blen, const char **array)

Append an array of null-terminated strings to a dynamically allocated buffer for serialization.

- size_t [_mem_append_serialized_array_cb](#) (unsigned char **buf, size_t *blen, const char **array, [custom_serializer_t](#) sf)

Append an array of custom-formatted data to a dynamically allocated buffer for serialization.

- unsigned int [_deserialize_data](#) (unsigned char *dst, unsigned char **bufptr, const unsigned char *bufend, size_t len)

Deserialize a fixed-sized chunk of data from an input buffer.

- unsigned char * [_deserialize_vardata](#) (unsigned char **bufptr, const unsigned char *bufend, size_t *outlen)
Deserialize a variable-length chunk of data from an input buffer.
- unsigned int [_deserialize_string](#) (char **dst, unsigned char **bufptr, const unsigned char *bufend)
Deserialize a null-terminated string from an input buffer.
- unsigned int [_deserialize_array](#) (unsigned char ***dst, unsigned char **bufptr, const unsigned char *bufend, size_t itemsize)
Deserialize an array of fixed-size objects from an input buffer.
- unsigned int [_deserialize_str_array](#) (char ***dst, unsigned char **bufptr, const unsigned char *bufend)
Deserialize an array of null-terminated string from an input buffer.
- unsigned int [_deserialize_array_cb](#) (unsigned char ***dst, unsigned char **bufptr, const unsigned char *bufend, [custom_deserializer_t](#) dfn)
Deserialize an array of custom-formatted data objects from an input buffer using a deserialization function.
- int [_is_object_expired](#) ([cached_object_t](#) *obj, int *refresh)
Check to see if a cached object needs to be evicted for exceeding its TTL or expiration.
- [cached_object_t](#) * [_unlink_object](#) ([cached_object_t](#) *object, int destroy, int stale)
Unlink a cached object from its doubly linked list.
- [cached_object_t](#) * [_replace_object](#) ([cached_object_t](#) *oobj, [cached_object_t](#) *nobj, int shadow)
Replace one object in the cache with another.
- unsigned int [_evict_if_stale](#) ([cached_object_t](#) **objptr)
Evict an item from the object cache if it is stale (has expired).
- void [_lock_cache_store](#) ([cached_store_t](#) *store)
Lock a cached store for thread-safe processing.
- void [_unlock_cache_store](#) ([cached_store_t](#) *store)
Unlock a cached store for use by other callers.
- void * [_deserialize_signet_cb](#) (void *data, size_t len)
Deserializes signet from a void pointer.
- void * [_serialize_signet_cb](#) (void *record, size_t *outlen)
Serializes a [signet_t](#) structure into a binary string.
- void [_destroy_signet_cb](#) (void *record)
Destroys a [signet_t](#) structure.
- void [_dump_signet_cb](#) (FILE *fp, void *record, int brief)
Dumps a signet from a [signet_t](#) structure.

Variables

- [cached_store_t](#) [cached_stores](#) []

5.93.1 Define Documentation

5.93.1.1 `#define CACHE_PERM_ADD 0x8`

Definition at line 17 of file cache.h.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, and `_add_cached_object_forced()`.

5.93.1.2 `#define CACHE_PERM_ALL_FLAGS (CACHE_PERM_LOAD | CACHE_PERM_SAVE | CACHE_PERM_READ | CACHE_PERM_ADD | CACHE_PERM_DELETE)`

Definition at line 21 of file cache.h.

Referenced by `_set_cache_permissions()`.

5.93.1.3 `#define CACHE_PERM_DEFAULT CACHE_PERM_ALL_FLAGS`

Definition at line 22 of file cache.h.

5.93.1.4 `#define CACHE_PERM_DELETE 0x16`

Definition at line 18 of file cache.h.

Referenced by `_remove_cached_object()`, and `_remove_cached_object_cmp()`.

5.93.1.5 `#define CACHE_PERM_LOAD 0x1`

Definition at line 14 of file cache.h.

Referenced by `_load_cache_contents()`, and `_save_cache_contents()`.

5.93.1.6 `#define CACHE_PERM_NONE 0`

Definition at line 20 of file cache.h.

5.93.1.7 `#define CACHE_PERM_READ 0x4`

Definition at line 16 of file cache.h.

Referenced by `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_find_cached_object()`, and `_find_cached_object_cmp()`.

5.93.1.8 `#define CACHE_PERM_SAVE 0x2`

Definition at line 15 of file cache.h.

5.93.2 Typedef Documentation

5.93.2.1 `typedef struct cached_object cached_object_t`

5.93.2.2 `typedef int(* cached_store_comparator_t)(const void *, const void *)`

Definition at line 67 of file cache.h.

5.93.2.3 `typedef unsigned int(* custom_deserializer_t)(unsigned char **, unsigned char **, const unsigned char *)`

Definition at line 68 of file `cache.h`.

5.93.2.4 `typedef void(* custom_dumper_t)(FILE *, void *, int)`

Definition at line 71 of file `cache.h`.

5.93.2.5 `typedef size_t(* custom_serializer_t)(unsigned char **, size_t *, const void *)`

Definition at line 69 of file `cache.h`.

5.93.3 Enumeration Type Documentation

5.93.3.1 `enum cached_data_type_t`

Enumerator:

cached_data_unknown
cached_data_drec
cached_data_dnskey
cached_data_ds
cached_data_ocsp
cached_data_signet

Definition at line 25 of file `cache.h`.

5.93.4 Function Documentation

5.93.4.1 `cached_object_t* _clone_cached_object (const cached_object_t * obj)`

Clone a deep copy of a cached object for user-safe retrieval.

Definition at line 994 of file `cache.c`.

References `_get_cached_store_by_type()`, `cached_store_t::clone`, `cached_object::data`, `data`, `cached_store_t::deserialize`, `cached_object::dtype`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `cached_object::expiration`, `cached_object::id`, `cached_store_t::internal`, `cached_object::persists`, `PUSH_ERROR_SYSCALL`, `cached_object::relaxed`, `RET_ERROR_PTR`, `cached_store_t::serialize`, `store`, `cached_object::timestamp`, and `cached_object::ttl`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_find_cached_object()`, and `_find_cached_object_cmp()`.

5.93.4.2 `cached_object_t* _create_cached_object (cached_data_type_t dtype, unsigned long ttl, time_t expiration, void * data, int persists, int relaxed)`

Allocate and initialize a new cached object.

See also:

[_add_cached_object\(\)](#)

Note:

The `data` parameter specifies an object-specific value that will be passed to other parts of the cache management subsystem, such as functions for deletion, destruction, search, etc.

Parameters:

dtype the data type identifying the cached store to which this cached object will belong.

ttl if not 0, an optional time-to-live value in seconds specifying in how many seconds from now the cached object will reach expiration.

expiration if not 0, an optional UTC time value specifying the exact time at which this cached object will reach expiration.

data a pointer to the object-specific data to be associated with the cache entry.

persists if set, the cached object will be persisted to disk when the cache is saved.

relaxed if set, use a relaxed cache policy. A relaxed cache policy ensures that even if the entry's ttl has expired, if its absolute (UTC) expiration time has not been reached, it will not be evicted from the cache, but will allow for a notification to the caller that it is time for the cached entry to be refreshed.

Returns:

a pointer to a newly allocated cached object for the specified data on success, or NULL on failure.

Definition at line 244 of file cache.c.

References `cached_object::data`, `cached_object::dtype`, `ERR_NOMEM`, `ERR_UNSPEC`, `cached_object::expiration`, `cached_object::persists`, `PUSH_ERROR_SYSCALL`, `cached_object::relaxed`, `RET_ERROR_PTR`, `cached_object::timestamp`, and `cached_object::ttl`.

Referenced by `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, and `_get_dime_record()`.

5.93.4.3 `unsigned int _deserialize_array (unsigned char *** dst, unsigned char ** bufptr, const unsigned char * bufend, size_t itemsz)`

Deserialize an array of fixed-size objects from an input buffer.

Note:

This is the inverse function of [_mem_append_serialized_array\(\)](#).

Parameters:

dst a pointer to the address of an array of objects that will receive a copy of the deserialized data as an array of (NULL pointer terminated) fixed-size objects on success.

bufptr a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.

bufend a pointer to the end of the input buffer for validation purposes.

itemsz the size, in bytes, of the fixed-sized objects to be deserialized.

Returns:

1 on success or 0 on failure.

Definition at line 1876 of file cache.c.

References `_deserialize_data()`, `_ptr_chain_add()`, `_ptr_chain_free()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_UINT`.

Referenced by `_deserialize_dime_record_cb()`.

5.93.4.4 `unsigned int _deserialize_array_cb (unsigned char *** dst, unsigned char ** bufptr, const unsigned char * bufend, custom_deserializer_t dfn)`

Deserialize an array of custom-formatted data objects from an input buffer using a deserialization function.

Note:

This is the inverse function of [_mem_append_serialized_array_cb\(\)](#).

Parameters:

- dst* a pointer to the address of an array of data buffers that will receive a copy of the deserialized data as an array of (NULL pointer terminated) data buffers on success.
- bufptr* a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.
- bufend* a pointer to the end of the input buffer for validation purposes.
- dfn* a custom deserialization function that will be used to deserialize the user data from the input buffer.

Returns:

- 1 on success or 0 on failure.

Definition at line 2025 of file cache.c.

References `_deserialize_data()`, `_ptr_chain_add()`, `_ptr_chain_free()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, and `RET_ERROR_UINT`.

5.93.4.5 `unsigned int _deserialize_data (unsigned char * dst, unsigned char ** bufptr, const unsigned char * bufend, size_t len)`

Deserialize a fixed-sized chunk of data from an input buffer.

Note:

- This is the inverse function of `_mem_append()`; *dst* must be at least *len* bytes long.

Parameters:

- dst* a pointer to the output buffer that will receive the deserialized data.
- bufptr* a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.
- bufend* a pointer to the end of the input buffer for validation purposes.
- len* the size of the data chunk to be deserialized from the input buffer.

Returns:

- 1 on success or 0 on failure.

Definition at line 1755 of file cache.c.

References `ERR_BAD_PARAM`, `ERR_UNSPEC`, `RET_ERROR_UINT`, and `RET_ERROR_UINT_FMT`.

Referenced by `_deserialize_array()`, `_deserialize_array_cb()`, `_deserialize_dime_record_cb()`, `_deserialize_dnskey_record_cb()`, `_deserialize_ds_record_cb()`, `_deserialize_str_array()`, and `_deserialize_vardata()`.

5.93.4.6 `void* _deserialize_signet_cb (void * data, size_t len)`

Deserializes signet from a void pointer.

Parameters:

- data* Pointer to serial signet data.
- len* Length of serial signet data.

Returns:

- Void pointer to a `signet_t` structure.

Definition at line 2321 of file cache.c.

References `dime_sgmt_signet_binary_deserialize()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, and `RET_ERROR_PTR`.

5.93.4.7 unsigned int _deserialize_str_array (char * *dst*, unsigned char ** *bufptr*, const unsigned char * *bufend*)**

Deserialize an array of null-terminated string from an input buffer.

Note:

This is the inverse function of [_mem_append_serialized_str_array\(\)](#).

Parameters:

dst a pointer to the address of an array of strings that will receive a copy of the deserialized data as an array of (NULL pointer terminated) null-terminated strings on success.

bufptr a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.

bufend a pointer to the end of the input buffer for validation purposes.

Returns:

1 on success or 0 on failure.

Definition at line 1955 of file cache.c.

References [_deserialize_data\(\)](#), [_deserialize_string\(\)](#), [_ptr_chain_add\(\)](#), [_ptr_chain_free\(\)](#), [ERR_BAD_PARAM](#), [ERR_UNSPEC](#), and [RET_ERROR_UINT](#).

Referenced by [_deserialize_dime_record_cb\(\)](#).

5.93.4.8 unsigned int _deserialize_string (char ** *dst*, unsigned char ** *bufptr*, const unsigned char * *bufend*)

Deserialize a null-terminated string from an input buffer.

Note:

This is the inverse function of [_mem_append_serialized_string\(\)](#).

Parameters:

dst a pointer to the address of a string that will receive a copy of the deserialized data as a null-terminated string on success.

bufptr a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.

bufend a pointer to the end of the input buffer for validation purposes.

Returns:

1 on success or 0 on failure.

Definition at line 1826 of file cache.c.

References [ERR_BAD_PARAM](#), [ERR_UNSPEC](#), [PUSH_ERROR_SYSCALL](#), and [RET_ERROR_UINT](#).

Referenced by [_deserialize_dime_record_cb\(\)](#), [_deserialize_dnskey_record_cb\(\)](#), [_deserialize_ds_record_cb\(\)](#), and [_deserialize_str_array\(\)](#).

5.93.4.9 unsigned char* _deserialize_vardata (unsigned char ** *bufptr*, const unsigned char * *bufend*, size_t * *olen*)

Deserialize a variable-length chunk of data from an input buffer.

Note:

This is the inverse function of [_mem_append_serialized](#).

Parameters:

- bufptr* a pointer to the address of the input buffer holding the data to be deserialized, which will be updated after the operation to point to the next available data.
- bufend* a pointer to the end of the input buffer for validation purposes.
- olen* an optional pointer to receive the size of the deserialized data on success.

Returns:

a pointer to a newly allocated buffer containing the deserialized data on success, or NULL on failure.

Definition at line 1784 of file cache.c.

References `_deserialize_data()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, and `RET_ERROR_UINT`.

Referenced by `_deserialize_dnskey_record_cb()`, and `_deserialize_ds_record_cb()`.

5.93.4.10 void _destroy_signet_cb (void *record)

Destroys a `signet_t` structure.

Parameters:

- record* Void pointer to a `signet_t` structure to be destroyed.

Definition at line 2399 of file cache.c.

References `dime_sgmt_signet_destroy()`.

5.93.4.11 void _dump_cache (cached_data_type_t dtype, int do_data, int ephemeral)

Dump the content of the specified data cached store(s) to the console.

Parameters:

- dtype* the type of the object cached store to be dumped, or `cached_data_unknown` for all stores.
- do_data* if non-zero, dump the type-specific data associated with the cached object.
- ephemeral* if non-zero, print out ephemeral as well as persistent cache data.

Definition at line 1068 of file cache.c.

References `_dump_cache_data()`, `_get_chr_date()`, `_lock_cache_store()`, `_unlock_cache_store()`, `cached_data_unknown`, `cached_object::data`, `cached_store_t::dtype`, `cached_object::expiration`, `cached_object::next`, `cached_object::persists`, `cached_object::relaxed`, `cached_object::shadow`, `cached_object::timestamp`, and `cached_object::ttl`.

5.93.4.12 void _dump_cache_data (FILE *fp, const cached_object_t *obj, int brief)

Dump information about a specified cached object to a file stream.

Parameters:

- fp* a pointer to the file stream across which the data should be dumped.
- obj* a pointer to the cached object to have its details dumped.
- brief* if true, print a brief one-line description of the object; otherwise, dump full object information.

Definition at line 1163 of file cache.c.

References `_clear_error_stack()`, `_get_cached_store_by_type()`, `cached_object::data`, `cached_object::dtype`, `cached_store_t::dump`, `dump_error_stack()`, `get_last_error()`, and `store`.

Referenced by `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_dump_cache()`, `_load_cache_contents()`, and `_unlink_object()`.

5.93.4.13 `void _dump_signet_cb (FILE *fp, void *record, int brief)`

Dumps a signet from a `signet_t` structure.

Parameters:

fp File descriptor that specifies the output destination.

record Void pointer to a `signet_t` structure.

brief TODO

Definition at line 2377 of file cache.c.

References `dime_sgmt_signet_dump()`.

5.93.4.14 `unsigned int _evict_if_stale (cached_object_t **objptr)`

Evict an item from the object cache if it is stale (has expired).

Returns:

1 if the object was evicted for being stale or 0 if it was not.

Definition at line 2287 of file cache.c.

References `_clear_error_stack()`, `_is_object_expired()`, `_unlink_object()`, `dump_error_stack()`, and `get_last_error()`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_remove_cached_object()`, and `_remove_cached_object_cmp()`.

5.93.4.15 `cached_store_t* _get_cached_store_by_type (cached_data_type_t dtype)`

Get a pointer to a cached object store by its type.

Parameters:

dtype the numerical identifier of the cached data type.

Returns:

a pointer to the requested cached object store on success, or NULL on failure.

Definition at line 46 of file cache.c.

Referenced by `_clone_cached_object()`, `_destroy_cache_entry()`, `_dump_cache_data()`, `_get_cache_obj_data()`, `_load_cache_contents()`, `_replace_object()`, and `_unlink_object()`.

5.93.4.16 `int _is_object_expired (cached_object_t *obj, int *refresh)`

Check to see if a cached object needs to be evicted for exceeding its TTL or expiration.

Note:

An object has "expired" and needs to be evicted from the cache if it has a non-zero TTL from its cache date that has been reached, or a non-zero absolute expiration date that has been exceeded. If the object's cache policy is relaxed, a TTL expiration advises a "refresh" if no absolute expiration has occurred.

Parameters:

obj a pointer to the cached object to have its age examined.

refresh an optional pointer to a value that, if the cached object has a relaxed expiration policy, will be set to 1 if it needs to be refreshed or 0 if it does not.

Returns:

1 if the object has expired or 0 if it has not; -1 if a general error has occurred.

Definition at line 2093 of file cache.c.

References ERR_BAD_PARAM, ERR_UNSPEC, cached_object::expiration, PUSH_ERROR_SYSCALL, cached_object::relaxed, RET_ERROR_INT, cached_object::timestamp, and cached_object::ttl.

Referenced by _evict_if_stale(), and _get_dime_record().

5.93.4.17 void _lock_cache_store (cached_store_t * store)

Lock a cached store for thread-safe processing.

Parameters:

store a pointer to the cached store to be locked.

Definition at line 2261 of file cache.c.

References cached_store_t::lock.

Referenced by _add_cached_object(), _add_cached_object_cmp(), _cached_object_exists(), _cached_object_exists_cmp(), _dump_cache(), _find_cached_object(), _find_cached_object_cmp(), _fixup_dnskey_validation(), _load_cache_contents(), _remove_cached_object(), _remove_cached_object_cmp(), and _save_cache_contents().

5.93.4.18 size_t _mem_append_serialized (unsigned char ** buf, size_t * blen, const unsigned char * data, size_t dlen)

Append a variable-length chunk of data to a dynamically allocated buffer for serialization.

Parameters:

buf a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.

blen a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.

data a pointer to a buffer holding the data that will be appended to the end of the output buffer.

dlen the size, in bytes, of the data buffer to be appended to the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1524 of file cache.c.

References _mem_append(), ERR_BAD_PARAM, ERR_NOMEM, and RET_ERROR_UINT.

Referenced by _mem_append_serialized_array(), _mem_append_serialized_array_cb(), _mem_append_serialized_str_array(), _serialize_dnskey_record_cb(), and _serialize_ds_record_cb().

5.93.4.19 size_t _mem_append_serialized_array (unsigned char ** *buf*, size_t * *blen*, const unsigned char ** *array*, size_t *itemsz*)

Append an array of fixed-size objects to a dynamically allocated buffer for serialization.

Parameters:

- buf* a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.
- blen* a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.
- array* an array of pointers to fixed-size objects (ending with a NULL pointer) to be appended to the end of the output buffer.
- itemsz* the size, in bytes, of the array elements to be deserialized.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1585 of file cache.c.

References `_mem_append()`, `_mem_append_serialized()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

Referenced by `_serialize_dime_record_cb()`.

5.93.4.20 size_t _mem_append_serialized_array_cb (unsigned char ** *buf*, size_t * *blen*, const char ** *array*, custom_serializer_t *sfn*)

Append an array of custom-formatted data to a dynamically allocated buffer for serialization.

Note:

This function uses a custom serialization function in order to output the user data properly.

Parameters:

- buf* a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.
- blen* a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.
- array* an array of pointers to custom-formatted data buffers (ending with a NULL pointer) to be appended to the end of the output buffer using a custom serialization function.
- sfn* a custom serialization function that will be used to serialize the user data into the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1698 of file cache.c.

References `_mem_append()`, `_mem_append_serialized()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

5.93.4.21 size_t _mem_append_serialized_str_array (unsigned char ** *buf*, size_t * *blen*, const char ** *array*)

Append an array of null-terminated strings to a dynamically allocated buffer for serialization.

Parameters:

- buf* a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.
- blen* a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.
- array* an array of pointers to null-terminated strings (ending with a NULL pointer) to be appended to the end of the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1640 of file cache.c.

References `_mem_append()`, `_mem_append_serialized()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

Referenced by `_serialize_dime_record_cb()`.

5.93.4.22 `size_t _mem_append_serialized_string (unsigned char ** buf, size_t * blen, const char * string)`

Appended a null-terminated string to a dynamically allocated buffer for serialization.

Parameters:

- buf* a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.
- blen* a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.
- string* a null-terminated string that will be appended to the end of the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 1552 of file cache.c.

References `_mem_append()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, and `RET_ERROR_UINT`.

Referenced by `_serialize_dime_record_cb()`, `_serialize_dnskey_record_cb()`, and `_serialize_ds_record_cb()`.

5.93.4.23 `cached_object_t* _replace_object (cached_object_t * oobj, cached_object_t * nobj, int shadow)`

Replace one object in the cache with another.

Note:

If shadow is not set, then the old cached object will be automatically destroyed. Otherwise this function also makes the old cached object the new cached object's "shadow" data.

Parameters:

- oobj* a pointer to the old cached object to be replaced in its cached store with the new object.
- nobj* a pointer to the new cached object that will replace the old object in the cached store.
- shadow* if set, preserve the old object as the "shadow" value of the new object.

Returns:

NULL on failure, or a pointer to the new cached object on success.

Definition at line 2208 of file cache.c.

References `_destroy_cache_entry()`, `_get_cached_store_by_type()`, `cached_object::dtype`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `cached_store_t::head`, `cached_object::id`, `cached_object::next`, `cached_object::prev`, `RET_ERROR_PTR`, `SHA_256_SIZE`, `cached_object::shadow`, and `store`.

Referenced by `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, and `_get_dime_record()`.

5.93.4.24 `void* _serialize_signet_cb (void * record, size_t * outlen)`

Serializes a [signet_t](#) structure into a binary string.

Parameters:

- record* Void pointer to a [signet_t](#) structure to be serialized.

outlen Pointer to the length of the returned string.

Returns:

Pointer to a serialized signet.

Definition at line 2343 of file cache.c.

References dime_sgmt_signet_binary_serialize(), ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

5.93.4.25 cached_object_t* _unlink_object (cached_object_t * object, int destroy, int stale)

Unlink a cached object from its doubly linked list.

Parameters:

object a pointer to the cached object to be delinked.

destroy if set, deallocate the specified cached object after unlinking.

stale if set, the cache removal was performed because of a stale entry.

Returns:

a pointer to the next cached object in the store, or NULL if at the end of the store.

Definition at line 2140 of file cache.c.

References _clear_error_stack(), _dbgprint(), _dump_cache_data(), _get_cached_store_by_type(), _verbose, cached_object::data, cached_store_t::destructor, cached_object::dtype, cached_store_t::dump, dump_error_stack(), ERR_BAD_PARAM, ERR_UNSPEC, get_last_error(), cached_store_t::head, cached_object::next, cached_object::prev, RET_ERROR_PTR, and store.

Referenced by _do_ocsp_validation(), _evict_if_stale(), _remove_cached_object(), and _remove_cached_object_cmp().

5.93.4.26 void _unlock_cache_store (cached_store_t * store)

Unlock a cached storea for use by other callers.

Parameters:

store a pointer to the cached store to be unlocked.

Definition at line 2274 of file cache.c.

References cached_store_t::lock.

Referenced by _add_cached_object(), _add_cached_object_cmp(), _cached_object_exists(), _cached_object_exists_cmp(), _dump_cache(), _find_cached_object(), _find_cached_object_cmp(), _fixup_dnskey_validation(), _load_cache_contents(), _remove_cached_object(), _remove_cached_object_cmp(), and _save_cache_contents().

- 5.93.4.27 PUBLIC_FUNC_DECL (void *, get_cache_obj_data, cached_object_t * *object*)
- 5.93.4.28 PUBLIC_FUNC_DECL (void, destroy_cache_entry, cached_object_t * *entry*)
- 5.93.4.29 PUBLIC_FUNC_DECL (int, remove_cached_object_cmp, const void * *key*, cached_store_t * *store*, cached_store_comparator_t *cmpfn*)
- 5.93.4.30 PUBLIC_FUNC_DECL (int, remove_cached_object, const char * *oid*, cached_store_t * *store*)
- 5.93.4.31 PUBLIC_FUNC_DECL (cached_object_t *, add_cached_object_cmp_forced, const char * *id*, const void * *key*, cached_store_t * *store*, unsigned long *ttl*, time_t *expiration*, void * *data*, int *persistent*, int *relaxed*, cached_store_comparator_t *cmpfn*)
- 5.93.4.32 PUBLIC_FUNC_DECL (cached_object_t *, add_cached_object_forced, const char * *id*, cached_store_t * *store*, unsigned long *ttl*, time_t *expiration*, void * *data*, int *persistent*, int *relaxed*)
- 5.93.4.33 PUBLIC_FUNC_DECL (cached_object_t *, add_cached_object_cmp, const char * *id*, const void * *key*, cached_store_t * *store*, unsigned long *ttl*, time_t *expiration*, void * *data*, int *persistent*, int *relaxed*, cached_store_comparator_t *cmpfn*)
- 5.93.4.34 PUBLIC_FUNC_DECL (cached_object_t *, add_cached_object, const char * *id*, cached_store_t * *store*, unsigned long *ttl*, time_t *expiration*, void * *data*, int *persistent*, int *relaxed*)
- 5.93.4.35 PUBLIC_FUNC_DECL (int, cached_object_exists_cmp, const void * *key*, cached_store_t * *store*, cached_store_comparator_t *cmpfn*)
- 5.93.4.36 PUBLIC_FUNC_DECL (int, cached_object_exists, const unsigned char * *hashid*, cached_store_t * *store*)
- 5.93.4.37 PUBLIC_FUNC_DECL (cached_object_t *, find_cached_object_cmp, const void * *key*, cached_store_t * *store*, cached_store_comparator_t *cmpfn*)
- 5.93.4.38 PUBLIC_FUNC_DECL (cached_object_t *, find_cached_object, const char * *oid*, cached_store_t * *store*)
- 5.93.4.39 PUBLIC_FUNC_DECL (int, set_cache_permissions, unsigned long *flags*)
- 5.93.4.40 PUBLIC_FUNC_DECL (int, set_cache_location, const char * *path*)
- 5.93.4.41 PUBLIC_FUNC_DECL (char *, get_cache_location, void)
- 5.93.4.42 PUBLIC_FUNC_DECL (char *, get_dime_dir_location, const char * *suffix*)
- 5.93.4.43 PUBLIC_FUNC_DECL (int, save_cache_contents, void)
- 5.93.4.44 PUBLIC_FUNC_DECL (int, load_cache_contents, void)

5.93.5 Variable Documentation

5.93.5.1 cached_store_t cached_stores[]

Definition at line 26 of file cache.c.

Referenced by _add_dnskey_entry_rsa(), _add_ds_entry(), _do_ocsp_validation(), _fixup_dnskey_validation(), _get_dime_record(), _get_dime_record_from_file(), _get_dnskey_by_tag(), _get_ds_by_dnskey(), _get_signet(), and _load_dnskey_file().

5.94 src/engine/config/cache/keys.h File Reference

Variables

- [cache_keys_t cache_keys](#) []

5.94.1 Variable Documentation

5.94.1.1 [cache_keys_t cache_keys](#) []

Initial value:

```
{
{
    .offset = offsetof (cache_t, name),
    .norm.type = M_TYPE_NULLER,
    .norm.val.st = NULL,
    .name = ".name",
    .description = "The host name or address of the cache instance.",
    .required = true
},
{
    .offset = offsetof (cache_t, port),
    .norm.type = M_TYPE_UINT32,
    .norm.val.u32 = 11211,
    .name = ".port",
    .description = "The port used by the cache instance.",
    .required = false
},
{
    .offset = offsetof (cache_t, weight),
    .norm.type = M_TYPE_UINT32,

    .norm.val.u32 = 1024,
    .name = ".weight",
    .description = "The relative weight of the cache instance.",
    .required = false
}
}
```

Definition at line 11 of file keys.h.

Referenced by [cache_alloc\(\)](#), [cache_config\(\)](#), [cache_free\(\)](#), [cache_output_help\(\)](#), [cache_output_settings\(\)](#), and [cache_validate\(\)](#).

5.95 src/engine/config/global/keys.h File Reference

Variables

- [magma_keys_t magma_keys](#) []

5.95.1 Variable Documentation

5.95.1.1 magma_keys_t magma_keys[]

Definition at line 28 of file keys.h.

Referenced by `config_free()`, `config_key_lookup()`, `config_load_defaults()`, `config_output_help()`, `config_output_settings()`, and `config_validate_settings()`.

5.96 src/engine/config/relay/keys.h File Reference

Variables

- [relay_keys_t relay_keys \[\]](#)

5.96.1 Variable Documentation

5.96.1.1 relay_keys_t relay_keys[]

Initial value:

```
{
{
    .offset = offsetof (relay_t, name),
    .norm.type = M_TYPE_NULLER,
    .norm.val.st = NULL,
    .name = ".name",
    .description = "The host name or address of the mail relay server.",
    .required = true
},
{
    .offset = offsetof (relay_t, port),
    .norm.type = M_TYPE_UINT32,
    .norm.val.u32 = 25,
    .name = ".port",
    .description = "The port used by the mail relay server.",
    .required = false
},
{
    .offset = offsetof (relay_t, secure),
    .norm.type = M_TYPE_BOOLEAN,
    .norm.val.binary = false,
    .name = ".secure",
    .description = "Determines whether connections should be made using TLS.",
    .required = false
}
}
```

Definition at line 11 of file keys.h.

Referenced by relay_alloc(), relay_config(), relay_free(), relay_output_help(), relay_output_settings(), and relay_validate().

5.97 src/engine/config/servers/keys.h File Reference

Variables

- [server_keys_t server_keys](#) []

5.97.1 Variable Documentation

5.97.1.1 [server_keys_t server_keys](#) []

Definition at line 11 of file keys.h.

Referenced by [servers_alloc\(\)](#), [servers_config\(\)](#), [servers_free\(\)](#), [servers_output_help\(\)](#), [servers_output_settings\(\)](#), and [servers_validate\(\)](#).

5.98 src/providers/dime/signet/keys.h File Reference

```
#include "dime/signet/common.h"
```

Enumerations

- enum [keys_type_t](#) { [KEYS_TYPE_ERROR](#) = 0, [KEYS_TYPE_ORG](#), [KEYS_TYPE_USER](#) }

Functions

- int [dime_keys_file_create](#) ([keys_type_t](#) type, [ED25519_KEY](#) *sign_key, [EC_KEY](#) *enc_key, const char *filename)
Creates a keys file with specified signing and encryption keys.
- [EC_KEY](#) * [dime_keys_enckey_fetch](#) (const char *filename)
Retrieves the signing key from the keys file.
- [ED25519_KEY](#) * [dime_keys_signkey_fetch](#) (const char *filename)
Retrieves the encryption key from the keys file.
- int [dime_keys_generate](#) ([keys_type_t](#) type, char **signet_pem, char **key_pem)

5.98.1 Enumeration Type Documentation

5.98.1.1 enum keys_type_t

Enumerator:

KEYS_TYPE_ERROR
KEYS_TYPE_ORG
KEYS_TYPE_USER

Definition at line 6 of file keys.h.

5.98.2 Function Documentation

5.98.2.1 [EC_KEY](#)* [dime_keys_enckey_fetch](#) (char const * *filename*)

Retrieves the signing key from the keys file.

Parameters:

filename Null terminated filename string.

Returns:

Pointer to the ed25519 signing key. {free_ed25519_key}

Definition at line 633 of file keys.c.

References [PUBLIC_FUNCTION_IMPLEMENT](#).

5.98.2.2 int dime_keys_file_create (keys_type_t type, ED25519_KEY * sign_key, EC_KEY * enc_key, const char * filename)

Creates a keys file with specified signing and encryption keys.

Parameters:

type Type of keys file, whether the keys correspond to a user or organizational signet.

sign_key Pointer to the specified ed25519 key, the private portion of which will be stored in the keys file as the signing key.

enc_key Pointer to the specified elliptic curve key, the private portion of which will be stored in the keys file as the encryption key.

filename Pointer to the NULL terminated string containing the filename for the keys file.

Returns:

0 on success, -1 on failure.

Definition at line 586 of file keys.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.98.2.3 int dime_keys_generate (keys_type_t type, char ** signet_pem, char ** key_pem)

Definition at line 637 of file keys.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.98.2.4 ED25519_KEY* dime_keys_signkey_fetch (char const * filename)

Retrieves the encryption key from the keys file.

Parameters:

filename Null terminated filename string.

Returns:

Pointer to the elliptic curve encryption key. {free_ec_key}

Definition at line 620 of file keys.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.99 src/providers/prime/keys/keys.h File Reference

Functions

- [stringer_t * user_encrypted_key_get \(stringer_t *key, prime_user_key_t *user, stringer_t *output\)](#)

users.c

- [prime_user_key_t * user_encrypted_key_set \(stringer_t *key, stringer_t *user\)](#)
- [prime_user_key_t * user_key_alloc \(void\)](#)
- [void user_key_free \(prime_user_key_t *user\)](#)
- [prime_user_key_t * user_key_generate \(void\)](#)
- [stringer_t * user_key_get \(prime_user_key_t *user, stringer_t *output\)](#)
- [size_t user_key_length \(prime_user_key_t *user\)](#)
- [prime_user_key_t * user_key_set \(stringer_t *user\)](#)
- [stringer_t * org_encrypted_key_get \(stringer_t *key, prime_org_key_t *org, stringer_t *output\)](#)

orgs.c

- [prime_org_key_t * org_encrypted_key_set \(stringer_t *key, stringer_t *org\)](#)
- [prime_org_key_t * org_key_alloc \(void\)](#)
- [void org_key_free \(prime_org_key_t *org\)](#)
- [prime_org_key_t * org_key_generate \(void\)](#)
- [stringer_t * org_key_get \(prime_org_key_t *org, stringer_t *output\)](#)
- [size_t org_key_length \(prime_org_key_t *org\)](#)
- [prime_org_key_t * org_key_set \(stringer_t *org\)](#)

5.99.1 Function Documentation

5.99.1.1 [stringer_t* org_encrypted_key_get \(stringer_t * key, prime_org_key_t * org, stringer_t * output\)](#)

orgs.c

Definition at line 142 of file *orgs.c*.

References [aes_artifact_encrypt\(\)](#), [org_key_get\(\)](#), and [st_free\(\)](#).

Referenced by [prime_key_encrypt\(\)](#).

5.99.1.2 [prime_org_key_t* org_encrypted_key_set \(stringer_t * key, stringer_t * org\)](#)

Definition at line 159 of file *orgs.c*.

References [aes_artifact_decrypt\(\)](#), [org_key_set\(\)](#), [prime_org_key_t](#), and [st_free\(\)](#).

Referenced by [prime_key_decrypt\(\)](#).

5.99.1.3 [prime_org_key_t* org_key_alloc \(void\)](#)

Definition at line 21 of file *orgs.c*.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_org_key_t](#).

Referenced by [org_key_set\(\)](#).

5.99.1.4 void org_key_free (prime_org_key_t * org)

Definition at line 10 of file orgs.c.

References ed25519_free(), mm_free(), and secp256k1_free().

Referenced by org_key_generate(), org_key_set(), and prime_free().

5.99.1.5 prime_org_key_t* org_key_generate (void)

Definition at line 35 of file orgs.c.

References ed25519_generate(), log_pedantic, mm_alloc(), org_key_free(), prime_org_key_t, and secp256k1_generate().

Referenced by prime_key_generate().

5.99.1.6 stringer_t* org_key_get (prime_org_key_t * org, stringer_t * output)

Definition at line 61 of file orgs.c.

References ED25519_KEY_PRIV_LEN, ed25519_private_get(), length, log_pedantic, MANAGEDBUF, org_key_length(), prime_field_write(), prime_header_org_key_write(), PRIME_ORG_KEY, SECP256K1_KEY_PRIV_LEN, secp256k1_private_get(), st_alloc(), st_avail_get(), st_cleanup, st_opt_get(), st_valid_destination(), st_wipe(), and st_write.

Referenced by org_encrypted_key_get(), and prime_get().

5.99.1.7 size_t org_key_length (prime_org_key_t * org)

Definition at line 52 of file orgs.c.

Referenced by org_key_get().

5.99.1.8 prime_org_key_t* org_key_set (stringer_t * org)

Definition at line 100 of file orgs.c.

References ed25519_private_set(), log_pedantic, org_key_alloc(), org_key_free(), prime_field_get(), prime_field_t, prime_object_free(), prime_object_t, PRIME_ORG_KEY, prime_org_key_t, prime_unpack(), and secp256k1_private_set().

Referenced by org_encrypted_key_set(), and prime_set().

5.99.1.9 stringer_t* user_encrypted_key_get (stringer_t * key, prime_user_key_t * user, stringer_t * output)

users.c

Definition at line 142 of file users.c.

References aes_artifact_encrypt(), st_free(), and user_key_get().

Referenced by prime_key_encrypt().

5.99.1.10 prime_user_key_t* user_encrypted_key_set (stringer_t * key, stringer_t * user)

Definition at line 159 of file users.c.

References aes_artifact_decrypt(), prime_user_key_t, st_free(), and user_key_set().

Referenced by prime_key_decrypt().

5.99.1.11 prime_user_key_t* user_key_alloc (void)

Definition at line 21 of file users.c.

References log_pedantic, mm_alloc(), mm_wipe(), and prime_user_key_t.

Referenced by user_key_set().

5.99.1.12 void user_key_free (prime_user_key_t * user)

Definition at line 10 of file users.c.

References ed25519_free(), mm_free(), and secp256k1_free().

Referenced by prime_free(), user_key_generate(), and user_key_set().

5.99.1.13 prime_user_key_t* user_key_generate (void)

Definition at line 35 of file users.c.

References ed25519_generate(), log_pedantic, mm_alloc(), prime_user_key_t, secp256k1_generate(), and user_key_free().

Referenced by prime_key_generate().

5.99.1.14 stringer_t* user_key_get (prime_user_key_t * user, stringer_t * output)

Definition at line 61 of file users.c.

References ED25519_KEY_PRIV_LEN, ed25519_private_get(), length, log_pedantic, MANAGEDBUF, prime_field_write(), prime_header_user_key_write(), PRIME_USER_KEY, SECP256K1_KEY_PRIV_LEN, secp256k1_private_get(), st_alloc(), st_avail_get(), st_cleanup, st_opt_get(), st_valid_destination(), st_wipe(), st_write, and user_key_length().

Referenced by prime_get(), and user_encrypted_key_get().

5.99.1.15 size_t user_key_length (prime_user_key_t * user)

Definition at line 52 of file users.c.

Referenced by user_key_get().

5.99.1.16 prime_user_key_t* user_key_set (stringer_t * user)

Definition at line 100 of file users.c.

References ed25519_private_set(), log_pedantic, prime_field_get(), prime_field_t, prime_object_free(), prime_object_t, prime_unpack(), PRIME_USER_KEY, prime_user_key_t, secp256k1_private_set(), user_key_alloc(), and user_key_free().

Referenced by prime_set(), and user_encrypted_key_set().

5.100 src/engine/config/config.h File Reference

```
#include "cache/cache.h"
#include "relay/relay.h"
#include "servers/servers.h"
#include "global/global.h"
```

Defines

- #define [MAGMA_HOSTNAME_MAX](#) _POSIX_HOST_NAME_MAX
- #define [MAGMA_FILEPATH_MAX](#) PATH_MAX
- #define [MAGMA_FILENAME_MAX](#) NAME_MAX
- #define [MAGMA_THREAD_STACK_SIZE](#) 1048576
- #define [MAGMA_THREAD_BUFFER_SIZE](#) 1024
- #define [MAGMA_WORKER_THREAD_LIMIT](#) 16384
- #define [MAGMA_CRYPTOGRAPHY_SEED_SIZE](#) 256
- #define [MAGMA_LOGS](#) "logs/"
- #define [MAGMA_RESOURCE_FONTS](#) "resources/fonts/"
- #define [MAGMA_RESOURCE_PAGES](#) "resources/pages/"
- #define [MAGMA_RESOURCE_VIRUS](#) "resources/virus/"
- #define [MAGMA_RESOURCE_LOCATION](#) "resources/location/"
- #define [MAGMA_RESOURCE_TEMPLATES](#) "resources/templates/"
- #define [MAGMA_LOCATION_CACHE](#) CONSTANT("disable")
- #define [MAGMA_CACHE_INSTANCES](#) 8
- #define [MAGMA_CACHE_SERVER_RETRY](#) 600
- #define [MAGMA_CACHE_SOCKET_TIMEOUT](#) 10
- #define [MAGMA_BLACKLIST_INSTANCES](#) 6
- #define [MAGMA_RELAY_INSTANCES](#) 8
- #define [MAGMA_SERVER_INSTANCES](#) 32
- #define [MAGMA_CONNECTION_BUFFER_SIZE](#) 8192
- #define [MAGMA_SMTP_MAX_HELO_SIZE](#) MAGMA_HOSTNAME_MAX
- #define [MAGMA_SMTP_MAX_ADDRESS_SIZE](#) 256
- #define [MAGMA_SMTP_RECIPIENT_LIMIT](#) 256
- #define [MAGMA_SMTP_RELAY_LIMIT](#) 256
- #define [MAGMA_SMTP_LINE_WRAP_LENGTH](#) 80
- #define [MAGMA_SMTP_MAX_MESSAGE_SIZE](#) 1073741824
- #define [CONFIG_CHECK_EXISTS](#)(option, ptype)
- #define [CONFIG_CHECK_FILE_READABLE](#)(x) CONFIG_CHECK_EXISTS(x,"Filename")
- #define [CONFIG_CHECK_DIR_READABLE](#)(x) CONFIG_CHECK_EXISTS(x,"Directory")
- #define [CONFIG_CHECK_READWRITE](#)(option, ptype)
- #define [CONFIG_CHECK_FILE_READWRITE](#)(x) CONFIG_CHECK_READWRITE(x,"Filename")
- #define [CONFIG_CHECK_DIR_READWRITE](#)(x) CONFIG_CHECK_READWRITE(x,"Directory")

5.100.1 Define Documentation

5.100.1.1 #define CONFIG_CHECK_DIR_READABLE(x) CONFIG_CHECK_EXISTS(x,"Directory")

Definition at line 94 of file config.h.

Referenced by config_validate_settings().

5.100.1.2 #define CONFIG_CHECK_DIR_READWRITE(x) CONFIG_CHECK_READWRITE(x,"Directory")

Definition at line 103 of file config.h.

Referenced by config_validate_settings().

5.100.1.3 #define CONFIG_CHECK_EXISTS(option, ptype)**Value:**

```
do { \
    if (option && !file_accessible(option)) { \
        log_critical(#ptype " specified in " #option " is not accessible: { path = %s\n", error = %s }", option, strerror_r(errno, bufptr, buflen)); \
        result = false; \
    } \
} while (0)
```

Definition at line 86 of file config.h.

5.100.1.4 #define CONFIG_CHECK_FILE_READABLE(x) CONFIG_CHECK_EXISTS(x,"Filename")

Definition at line 93 of file config.h.

Referenced by config_validate_settings().

5.100.1.5 #define CONFIG_CHECK_FILE_READWRITE(x) CONFIG_CHECK_READWRITE(x,"Filename")

Definition at line 102 of file config.h.

5.100.1.6 #define CONFIG_CHECK_READWRITE(option, ptype)**Value:**

```
do { \
    if (option && !file_readwritable(option)) { \
        log_critical(#ptype " specified in " #option " is not accessible for reading\nand writing: { path = %s, error = %s }", option, strerror_r(errno, bufptr, buflen)); \
        result = false; \
    } \
} while (0)
```

Definition at line 95 of file config.h.

5.100.1.7 #define MAGMA_BLACKLIST_INSTANCES 6

Definition at line 54 of file config.h.

Referenced by config_value_set().

5.100.1.8 #define MAGMA_CACHE_INSTANCES 8

Definition at line 45 of file config.h.

Referenced by cache_config(), cache_free(), cache_output_settings(), cache_start(), and cache_validate().

5.100.1.9 #define MAGMA_CACHE_SERVER_RETRY 600

Definition at line 48 of file config.h.

5.100.1.10 #define MAGMA_CACHE_SOCKET_TIMEOUT 10

Definition at line 51 of file config.h.

5.100.1.11 #define MAGMA_CONNECTION_BUFFER_SIZE 8192

Definition at line 63 of file config.h.

5.100.1.12 #define MAGMA_CRYPTOGRAPHY_SEED_SIZE 256

Definition at line 31 of file config.h.

5.100.1.13 #define MAGMA_FILENAME_MAX NAME_MAX

Definition at line 19 of file config.h.

5.100.1.14 #define MAGMA_FILEPATH_MAX PATH_MAX

Definition at line 16 of file config.h.

Referenced by args_parse(), process_find_pid(), process_kill(), and tank_start().

5.100.1.15 #define MAGMA_HOSTNAME_MAX _POSIX_HOST_NAME_MAX

Definition at line 13 of file config.h.

Referenced by process_start(), and warehouse_fetch_domains().

5.100.1.16 #define MAGMA_LOCATION_CACHE CONSTANT("disable")

Definition at line 42 of file config.h.

5.100.1.17 #define MAGMA_LOGS "logs/"

Definition at line 34 of file config.h.

5.100.1.18 #define MAGMA_RELAY_INSTANCES 8

Definition at line 57 of file config.h.

Referenced by relay_config(), relay_counter(), relay_free(), relay_output_settings(), relay_validate(), and smtp_client_connect().

5.100.1.19 #define MAGMA_RESOURCE_FONTS "resources/fonts/"

Definition at line 35 of file config.h.

5.100.1.20 #define MAGMA_RESOURCE_LOCATION "resources/location/"

Definition at line 38 of file config.h.

5.100.1.21 #define MAGMA_RESOURCE_PAGES "resources/pages/"

Definition at line 36 of file config.h.

5.100.1.22 #define MAGMA_RESOURCE_TEMPLATES "resources/templates/"

Definition at line 39 of file config.h.

5.100.1.23 #define MAGMA_RESOURCE_VIRUS "resources/virus/"

Definition at line 37 of file config.h.

5.100.1.24 #define MAGMA_SERVER_INSTANCES 32

Definition at line 60 of file config.h.

Referenced by net_listen(), net_trigger(), servers_config(), servers_encryption_start(), servers_encryption_stop(), servers_free(), servers_get_by_protocol(), servers_get_by_socket(), servers_get_count_using_port(), servers_network_start(), servers_network_stop(), servers_output_settings(), and servers_validate().

5.100.1.25 #define MAGMA_SMTP_LINE_WRAP_LENGTH 80

Definition at line 80 of file config.h.

5.100.1.26 #define MAGMA_SMTP_MAX_ADDRESS_SIZE 256

Definition at line 70 of file config.h.

5.100.1.27 #define MAGMA_SMTP_MAX_HELO_SIZE MAGMA_HOSTNAME_MAX

Definition at line 67 of file config.h.

5.100.1.28 #define MAGMA_SMTP_MAX_MESSAGE_SIZE 1073741824

Definition at line 83 of file config.h.

5.100.1.29 #define MAGMA_SMTP_RECIPIENT_LIMIT 256

Definition at line 73 of file config.h.

5.100.1.30 #define MAGMA_SMTP_RELAY_LIMIT 256

Definition at line 75 of file config.h.

5.100.1.31 #define MAGMA_THREAD_BUFFER_SIZE 1024

Definition at line 25 of file config.h.

5.100.1.32 #define MAGMA_THREAD_STACK_SIZE 1048576

Definition at line 22 of file config.h.

5.100.1.33 #define MAGMA_WORKER_THREAD_LIMIT 16384

Definition at line 28 of file config.h.

5.101 src/objects/config/config.h File Reference

Enumerations

- enum { [USER_CONF_STATUS_NONE](#) = 0, [USER_CONF_STATUS_CRITICAL](#) = 1 }

Functions

- struct [__attribute__](#) ((packed))
- [user_config_t](#) * [user_config_alloc](#) (uint64_t usernum)
config.c
- [user_config_t](#) * [user_config_create](#) (uint64_t usernum)
Create a new user config collection object for the specified user and populate it with config entries from the database.
- int_t [user_config_edit](#) ([user_config_t](#) *collection, [stringer_t](#) *key, [stringer_t](#) *value)
Change the value of, or delete a key from a user's config collection.
- [user_config_entry_t](#) * [user_config_entry_alloc](#) ([stringer_t](#) *key, [stringer_t](#) *value, uint64_t flags)
Create and initialize new user config entry object.
- void [user_config_entry_free](#) ([user_config_entry_t](#) *entry)
Destroy a user config entry.
- void [user_config_free](#) ([user_config_t](#) *collection)
Free a user config collection object.
- int_t [user_config_update](#) ([user_config_t](#) *collection)
Update a collection of user config options from the database, if necessary.
- int_t [user_config_delete](#) (uint64_t usernum, [stringer_t](#) *key)
datatier.c
- bool_t [user_config_fetch](#) ([user_config_t](#) *collection)
Fetch a user's config collection from the database.
- int_t [user_config_upsert](#) (uint64_t usernum, [stringer_t](#) *key, [stringer_t](#) *value, uint64_t flags)
Update a user config entry in the database, or insert it if it does not already exist.

Variables

- [user_config_entry_t](#)
- [user_config_t](#)

5.101.1 Enumeration Type Documentation

5.101.1.1 anonymous enum

Enumerator:

[USER_CONF_STATUS_NONE](#)

USER_CONF_STATUS_CRITICAL

Definition at line 17 of file config.h.

5.101.2 Function Documentation**5.101.2.1 struct __attribute__((packed)) [read]**

Definition at line 22 of file config.h.

References `inx_t`, and `__attribute__::usernum`.

5.101.2.2 user_config_t* user_config_alloc (uint64_t usernum)

config.c config.c

Parameters:

usernum the numerical user id for the requested user.

Returns:

NULL on failure or a pointer to the newly allocated user config collection object on success.

Definition at line 44 of file config.c.

References `align()`, `inx_alloc()`, `log_pedantic`, `M_INX_TREE`, `mm_alloc()`, `mm_free()`, `user_config_entry_free()`, and `user_config_t`.

Referenced by `user_config_create()`.

5.101.2.3 user_config_t* user_config_create (uint64_t usernum)

Create a new user config collection object for the specified user and populate it with config entries from the database.

Parameters:

usernum the numerical user id for the user to be queried.

Returns:

NULL on failure, or a pointer to the newly created user config collection object on success.

Definition at line 106 of file config.c.

References `log_pedantic`, `OBJECT_CONFIG`, `serial_get()`, `user_config_alloc()`, `user_config_fetch()`, `user_config_free()`, and `user_config_t`.

Referenced by `portal_endpoint_config_edit()`, and `portal_endpoint_config_load()`.

5.101.2.4 int_t user_config_delete (uint64_t usernum, stringer_t * key)

datatier.c datatier.c

Parameters:

usernum the numerical userid of the user to whom the requested config key belongs.

key a managed string containing the name of the config key to be deleted.

Returns:

1 if the config option was successfully deleted, 0 if the config key couldn't be found in the database, or -1 on general failure.

Definition at line 16 of file `datatier.c`.

References `log_check`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `st_length_int()`, and `stmt_exec_affected()`.

Referenced by `user_config_edit()`.

5.101.2.5 int_t user_config_edit (user_config_t * collection, stringer_t * key, stringer_t * value)

Change the value of, or delete a key from a user's config collection.

Parameters:

collection a pointer to a collection of user's config entries.

key a pointer to a managed string containing the name of the user config key to be modified.

value a pointer to a managed string containing the name of the new value of the key, or NULL if it is to be deleted.

Returns:

-1 on error or 1 on success.

Definition at line 160 of file `config.c`.

References `inx_delete()`, `inx_find()`, `inx_replace()`, `log_pedantic`, `M_TYPE_STRINGER`, `OBJECT_CONFIG`, `serial_increment()`, `multi_t::st`, `st_cmp_cs_eq()`, `st_empty`, `user_config_delete()`, `user_config_entry_alloc()`, `user_config_entry_free()`, `user_config_entry_t`, `user_config_upsert()`, and `multi_t::val`.

Referenced by `portal_endpoint_config_edit()`.

5.101.2.6 user_config_entry_t* user_config_entry_alloc (stringer_t * key, stringer_t * value, uint64_t flags)

Create and initialize new user config entry object.

Parameters:

key a managed string containing the key of the config entry object.

value a managed string containing the value of the config entry object.

flags a bitmask reflecting the flags of the config entry object.

Returns:

NULL on failure, or a pointer to the newly allocated and initialized user config entry object on success.

Definition at line 70 of file `config.c`.

References `align()`, `FOREIGNDATA`, `JOINTED`, `log_pedantic`, `mm_alloc()`, `mm_copy()`, `PLACER_T`, `placer_t`, `st_data_get()`, `st_length_get()`, `STACK`, and `user_config_entry_t`.

Referenced by `user_config_edit()`, and `user_config_fetch()`.

5.101.2.7 void user_config_entry_free (user_config_entry_t * entry)

Destroy a user config entry.

Parameters:

entry a pointer to the user config entry object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file config.c.

References mm_free().

Referenced by user_config_alloc(), user_config_edit(), and user_config_fetch().

5.101.2.8 bool_t user_config_fetch (user_config_t * collection)

Fetch a user's config collection from the database.

Parameters:

collection a pointer to the specified user config collection object (with usernum field set by the caller) to be populated from the database.

Returns:

true on success or false on failure.

Definition at line 96 of file datatier.c.

References inx_insert(), log_info, log_pedantic, M_TYPE_STRINGER, mm_wipe(), PLACER, res_field_block(), res_field_length(), res_field_uint64(), res_row_next(), res_table_free(), multi_t::st, stmt_get_result(), user_config_entry_alloc(), user_config_entry_free(), user_config_entry_t, and multi_t::val.

Referenced by user_config_create(), and user_config_update().

5.101.2.9 void user_config_free (user_config_t * collection)

Free a user config collection object.

Parameters:

collection a pointer to the user config collection object to be freed.

Returns:

This function returns no value.

Definition at line 29 of file config.c.

References inx_cleanup(), and mm_free().

Referenced by portal_endpoint_config_edit(), portal_endpoint_config_load(), and user_config_create().

5.101.2.10 int_t user_config_update (user_config_t * collection)

Update a collection of user config options from the database, if necessary.

Note:

This function is not currently being used anywhere. If this function returns -1, the config entries are left in an undefined state and should not be relied upon.

Parameters:

collection a pointer to a collection of user config entries to be checked for updates.

Returns:

1 if the collection is up-to-date, 0 if it was refreshed to update changes, or -1 if the refresh operation failed.

Definition at line 132 of file config.c.

References `inx_truncate()`, `OBJECT_CONFIG`, `serial_get()`, and `user_config_fetch()`.

5.101.2.11 int_t user_config_upsert (uint64_t usernum, stringer_t * key, stringer_t * value, uint64_t flags)

Update a user config entry in the database, or insert it if it does not already exist.

Parameters:

usernum the numerical id of the user to whom the specified config entry belongs.

key a pointer to a managed string containing the name of the config entry to be updated or inserted.

value a pointer to a managed string containing the value of the specified config entry key.

flags a bitmask of flags for the config entry (`USER_CONF_STATUS_CRITICAL` is supported).

Returns:

2 if the config entry was updated, 1 if a new config key was inserted into the database, 0 if no update was necessary, or -1 on general failure.

Definition at line 52 of file datatier.c.

References `log_check`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `st_length_int()`, and `stmt_exec_affected()`.

Referenced by `user_config_edit()`.

5.101.3 Variable Documentation**5.101.3.1 user_config_entry_t**

Definition at line 14 of file config.h.

Referenced by `portal_config_collection()`, `user_config_edit()`, `user_config_entry_alloc()`, and `user_config_fetch()`.

5.101.3.2 user_config_t

Definition at line 25 of file config.h.

Referenced by `portal_endpoint_config_edit()`, `portal_endpoint_config_load()`, `user_config_alloc()`, and `user_config_create()`.

5.102 src/engine/config/global/datatier.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t config_fetch_host_number (void)`
Retrieve this computer's host number from the database.
- `table_t * config_fetch_settings (void)`
Retrieve the entire collection of configuration key/value pairs from the database.

5.102.1 Function Documentation

5.102.1.1 `uint64_t config_fetch_host_number (void)`

Retrieve this computer's host number from the database. `datatier.c`

Returns:

this host's numerical identifier, or 0 on failure.

Definition at line 14 of file `datatier.c`.

References `magma_t::host`, `log_error`, `magma`, `mm_wipe()`, `magma_t::name`, `ns_length_get()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, and `stmt_get_result()`.

Referenced by `config_load_database_settings()`.

5.102.1.2 `table_t* config_fetch_settings (void)`

Retrieve the entire collection of configuration key/value pairs from the database.

Parameters:

none This function accepts no parameters.

Returns:

NULL on failure, or a database table of configuration key/value pairs on success.

Definition at line 40 of file `datatier.c`.

References `magma_t::host`, `magma`, `mm_wipe()`, `magma_t::name`, `ns_length_get()`, and `stmt_get_result()`.

Referenced by `config_load_database_settings()`.

5.103 src/objects/auth/datatier.c File Reference

```
#include "magma.h"
```

Functions

- `int_t auth_data_update_legacy` (uint64_t usernum, `stringer_t` *legacy, `stringer_t` *salt, `stringer_t` *verification, uint32_t bonus)
Replaces legacy auth tokens with STACIE compatible tokens.
- `void auth_data_update_lock` (uint64_t usernum, `auth_lock_status_t` lock)
Update a user lock status in the database.
- `int_t auth_data_fetch` (`auth_t` *auth)
Fetches the authentication information based on the provided username.

5.103.1 Function Documentation

5.103.1.1 int_t auth_data_fetch (auth_t * auth)

Fetches the authentication information based on the provided username. datatier.c

Note:

This function searches the Users table first, and the Mailboxes table second. This way if someone sneaks a username into the Mailboxes table, they won't override the Users table.

Parameters:

auth The authentication object providing the username, and used to store the result.

Returns:

0 if the user information is pulled correctly, 1 if the user isn't found, and -1 if an error occurs.

Definition at line 134 of file datatier.c.

References `base64_decode_mod()`, `auth_t::bonus`, `hex_decode_st()`, `auth_t::legacy`, `auth_t::locked`, `log_error`, `log_pedantic`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_int8()`, `res_field_length()`, `res_field_string()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_count()`, `res_row_next()`, `res_table_free()`, `auth_t::salt`, `auth_t::seasoning`, `st_char_get()`, `st_cmp_cs_eq()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_int()`, `st_search_chr()`, `STACIE_KEY_ROUNDS_MAX`, `STACIE_SALT_LENGTH`, `STACIE_TOKEN_LENGTH`, `auth_t::status`, `stmt_get_result()`, `auth_t::token`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, and `auth_t::verification`.

Referenced by `auth_challenge()`.

5.103.1.2 int_t auth_data_update_legacy (uint64_t usernum, stringer_t * legacy, stringer_t * salt, stringer_t * verification, uint32_t bonus)

Replaces legacy auth tokens with STACIE compatible tokens.

Parameters:

usernum The user account number.

legacy The legacy auth token, encoded as a hexadecimal string.

salt The user specific salt value, encoded as a hexadecimal string.

verification The STACIE verification token, encoded as a hexadecimal string.

bonus The number of bonus hash rounds.

Returns:

0 if the user information was updated correctly, or -1 if an error occurs.

Definition at line 22 of file `datatier.c`.

References `log_error`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_empty`, `st_length_get()`, `STACIE_KEY_ROUNDS_MAX`, and `stmt_exec_affected()`.

Referenced by `auth_login()`.

5.103.1.3 `void auth_data_update_lock (uint64_t usernum, auth_lock_status_t lock)`

Update a user lock status in the database.

Parameters:

usernum the numerical id of the user for whom the lock will be set.

lock the new value to which the specified user's lock will be set.

Returns:

This function returns no value.

Definition at line 89 of file `datatier.c`.

References `log_pedantic`, `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `auth_login()`.

5.104 src/objects/config/datatier.c File Reference

```
#include "magma.h"
```

Functions

- [int_t user_config_delete](#) (uint64_t usernum, [stringer_t](#) *key)
Delete a user config entry from the database by key.
- [int_t user_config_upsert](#) (uint64_t usernum, [stringer_t](#) *key, [stringer_t](#) *value, uint64_t flags)
Update a user config entry in the database, or insert it if it does not already exist.
- [bool_t user_config_fetch](#) ([user_config_t](#) *collection)
Fetch a user's config collection from the database.

5.104.1 Function Documentation

5.104.1.1 int_t user_config_delete (uint64_t usernum, stringer_t * key)

Delete a user config entry from the database by key. datatier.c

Parameters:

usernum the numerical userid of the user to whom the requested config key belongs.
key a managed string containing the name of the config key to be deleted.

Returns:

1 if the config option was successfully deleted, 0 if the config key couldn't be found in the database, or -1 on general failure.

Definition at line 16 of file datatier.c.

References [log_check](#), [log_pedantic](#), [mm_wipe\(\)](#), [st_char_get\(\)](#), [st_length_get\(\)](#), [st_length_int\(\)](#), and [stmt_exec_affected\(\)](#).

Referenced by [user_config_edit\(\)](#).

5.104.1.2 bool_t user_config_fetch (user_config_t * collection)

Fetch a user's config collection from the database.

Parameters:

collection a pointer to the specified user config collection object (with usernum field set by the caller) to be populated from the database.

Returns:

true on success or false on failure.

Definition at line 96 of file datatier.c.

References [inx_insert\(\)](#), [log_info](#), [log_pedantic](#), [M_TYPE_STRINGER](#), [mm_wipe\(\)](#), [PLACER](#), [res_field_block\(\)](#), [res_field_length\(\)](#), [res_field_uint64\(\)](#), [res_row_next\(\)](#), [res_table_free\(\)](#), [multi_t::st](#), [stmt_get_result\(\)](#), [user_config_entry_alloc\(\)](#), [user_config_entry_free\(\)](#), [user_config_entry_t](#), and [multi_t::val](#).

Referenced by [user_config_create\(\)](#), and [user_config_update\(\)](#).

5.104.1.3 `int_t user_config_upsert (uint64_t usernum, stringer_t * key, stringer_t * value, uint64_t flags)`

Update a user config entry in the database, or insert it if it does not already exist.

Parameters:

usernum the numerical id of the user to whom the specified config entry belongs.

key a pointer to a managed string containing the name of the config entry to be updated or inserted.

value a pointer to a managed string containing the value of the specified config entry key.

flags a bitmask of flags for the config entry (USER_CONF_STATUS_CRITICAL is supported).

Returns:

2 if the config entry was updated, 1 if a new config key was inserted into the database, 0 if no update was necessary, or -1 on general failure.

Definition at line 52 of file `datatier.c`.

References `log_check`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `st_length_int()`, and `stmt_exec_affected()`.

Referenced by `user_config_edit()`.

5.105 src/objects/contacts/datatier.c File Reference

```
#include "magma.h"
```

Functions

- [int_t contact_update_stamp](#) (uint64_t contactnum, uint64_t usernum, uint64_t foldernum)
Touch the last updated time stamp of a contact entry in the database.
- [int_t contact_delete](#) (uint64_t contactnum, uint64_t usernum, uint64_t foldernum)
Delete a user's contact entry and its associated details from the database.
- [int_t contact_update](#) (uint64_t contactnum, uint64_t usernum, uint64_t cur_folder, uint64_t target_folder, [stringer_t](#) *name)
Update a user contact entry in the database.
- uint64_t [contact_insert](#) (uint64_t usernum, uint64_t foldernum, [stringer_t](#) *name)
Insert a new contact entry into the database.
- [int_t contact_detail_delete](#) (uint64_t contactnum, [stringer_t](#) *key)
Delete the specified contact detail of a contact entry from the database.
- [int_t contact_detail_upsert](#) (uint64_t contactnum, [stringer_t](#) *key, [stringer_t](#) *value, uint64_t flags)
Update a specified contact detail in the database, or insert it if it does not exist.
- [int_t contact_details_fetch](#) ([contact_t](#) *contact)
Populate a contact entry with its details from the database.
- [int_t contacts_fetch](#) (uint64_t usernum, [contact_folder_t](#) *folder)
Retrieve all of a user's contact entries in a specified contacts folder from the database.

5.105.1 Function Documentation

5.105.1.1 int_t contact_delete (uint64_t contactnum, uint64_t usernum, uint64_t foldernum)

Delete a user's contact entry and its associated details from the database. datatier.c

Parameters:

- contactnum** the numerical id of the contact entry to be deleted.
- usernum** the numerical id of the user to whom the specified contact entry belongs.
- foldernum** the numerical id of the parent contact folder containing the contact entry.

Returns:

- 1 if the specified contact was deleted successfully, 0 if no matching contact was found in the database, or -1 on general failure.

Definition at line 61 of file datatier.c.

References [log_check](#), [log_pedantic](#), [mm_wipe\(\)](#), and [stmt_exec_affected\(\)](#).

Referenced by [contact_remove\(\)](#), and [portal_endpoint_contacts_remove\(\)](#).

5.105.1.2 `int_t contact_detail_delete (uint64_t contactnum, stringer_t * key)`

Delete the specified contact detail of a contact entry from the database.

Parameters:

contactnum the numerical id of the contact entry to have the specified detail removed.

key a managed string containing the name of the contact detail to be removed from the entry.

Returns:

-1 on error, 0 if no matching detail was found in the database, or 1 if the delete operation was successful.

Definition at line 204 of file `datatier.c`.

References `log_check`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `st_length_int()`, and `stmt_exec_affected()`.

Referenced by `contact_edit()`.

5.105.1.3 `int_t contact_detail_upsert (uint64_t contactnum, stringer_t * key, stringer_t * value, uint64_t flags)`

Update a specified contact detail in the database, or insert it if it does not exist.

Parameters:

contactnum the numerical id of the contact entry to be modified.

key a managed string containing the name of the contact detail to be updated.

value a managed string containing the new value of the specified contact detail.

flags a bitmask of flags to be associated with the specified contact entry detail.

Returns:

-1 on error, 0 if no update was necessary, 1 if a new contact detail was inserted into the database, or 2 if the specified contact detail was updated successfully.

Definition at line 241 of file `datatier.c`.

References `log_check`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `st_length_int()`, and `stmt_exec_affected()`.

Referenced by `contact_edit()`.

5.105.1.4 `int_t contact_details_fetch (contact_t * contact)`

Populate a contact entry with its details from the database.

Parameters:

contact a pointer to the contact entry object that will be updated.

Returns:

-1 on failure or 1 on success.

Definition at line 285 of file `datatier.c`.

References `contact_detail_alloc()`, `contact_detail_free()`, `contact_detail_t`, `inx_insert()`, `log_info`, `log_pedantic`, `M_TYPE_STRINGER`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `multi_t::st`, `stmt_get_result()`, and `multi_t::val`.

Referenced by `contacts_fetch()`.

5.105.1.5 uint64_t contact_insert (uint64_t usernum, uint64_t foldernum, stringer_t * name)

Insert a new contact entry into the database.

Parameters:

- usernum* the numerical id of the user to whom the contact entry belongs.
- foldernum* the numerical id of the parent folder to contain the contact entry.
- name* a pointer to a managed string containing the name of the new contact entry.

Returns:

- 1 on failure, 0 if no item was inserted into the database, or the id of the newly inserted contact entry in the database on success.

Definition at line 172 of file datatier.c.

References mm_wipe(), st_char_get(), st_length_get(), and stmt_insert().

Referenced by contact_create().

5.105.1.6 int_t contact_update (uint64_t contactnum, uint64_t usernum, uint64_t cur_folder, uint64_t target_folder, stringer_t * name)

Update a user contact entry in the database.

Parameters:

- contactnum* the numerical id of the contact entry to be modified.
- usernum* the numerical id of the user to whom the specified contact entry belongs.
- cur_folder* the numerical id of the parent contact containing the specified contact entry.
- target_folder* if not 0, sets the new parent contact folder to which the specified contact entry will belong.
- name* if not NULL, sets the new name of the specified contact entry.

Returns:

- 1 on error, 0 if the specified contact entry was not found in the database, or 1 if the contact entry was successfully updated.

Definition at line 106 of file datatier.c.

References ISNULL, log_check, log_pedantic, mm_wipe(), st_char_get(), st_empty, st_length_get(), and stmt_exec_affected().

Referenced by contact_edit(), and contact_move().

5.105.1.7 int_t contact_update_stamp (uint64_t contactnum, uint64_t usernum, uint64_t foldernum)

Touch the last updated time stamp of a contact entry in the database.

Parameters:

- contactnum* the numerical id of the contact entry to have its time stamp updated.
- usernum* the numerical id of the user to whom the contact entry belongs.
- foldernum* the numerical id of the parent folder containing the contact entry.

Returns:

- 1 if the contact time stamp was updated successfully, 0 if the specified contact was not found, or -1 on general failure.

Definition at line 17 of file datatier.c.

References log_check, log_pedantic, mm_wipe(), and stmt_exec_affected().

Referenced by contact_edit().

5.105.1.8 `int_t contacts_fetch(uint64_t usernum, contact_folder_t *folder)`

Retrieve all of a user's contact entries in a specified contacts folder from the database.

Parameters:

usernum the numerical id of the user whose contacts will be retrieved.

folder a pointer to the contact folder which will have its contents listed.

Returns:

-1 on failure or 1 on success.

LOW: Should we bother with error checking?

Definition at line 339 of file `datatier.c`.

References `contact_alloc()`, `contact_details_fetch()`, `contact_free()`, `contact_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_insert()`, `log_info`, `log_pedantic`, `M_TYPE_UINT64`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `contacts_update()`.

5.106 src/objects/folders/datatier.c File Reference

```
#include "magma.h"
```

Functions

- `inx_t * magma_folder_fetch` (uint64_t usernum, uint_t type)
Fetch all of a user's folders of a specified type from the database.
- `uint64_t magma_folder_insert` (uint64_t usernum, stringer_t *name, uint64_t parent, uint32_t order, uint_t type)
Insert a new folder into the database.
- `bool_t magma_folder_delete` (uint64_t usernum, uint64_t foldernum, uint_t type)
Delete a folder from the database.
- `bool_t magma_folder_rename` (uint64_t usernum, uint64_t foldernum, uint_t type, stringer_t *rename)
Rename a folder in the database.

5.106.1 Function Documentation

5.106.1.1 bool_t magma_folder_delete (uint64_t usernum, uint64_t foldernum, uint_t type)

Delete a folder from the database. datatier.c

Parameters:

- usernum** the numerical id of the user requesting the folder removal.
- foldernum** the numerical id of the folder to be removed.
- type** the type of the folder to be deleted (can be M_FOLDER_MESSAGES or M_FOLDER_CONTACTS).

Returns:

true on success or false on failure.

Definition at line 136 of file datatier.c.

References `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `contact_folder_remove()`, and `message_folder_remove()`.

5.106.1.2 inx_t* magma_folder_fetch (uint64_t usernum, uint_t type)

Fetch all of a user's folders of a specified type from the database.

Parameters:

- usernum** the numerical id of the requested user.
- type** the folder class of the folders to be retrieved (can be M_FOLDER_MESSAGES or M_FOLDER_CONTACTS).

Returns:

NULL on failure, or an inx object holding all of the requested folders on success.

Definition at line 16 of file `datatier.c`.

References `inx_alloc()`, `inx_free()`, `inx_insert()`, `inx_t`, `log_info`, `log_pedantic`, `M_INX_TREE`, `M_TYPE_UINT64`, `magma_folder_funcs()`, `magma_folder_t`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `contacts_update()`, and `messages_update()`.

5.106.1.3 `uint64_t magma_folder_insert(uint64_t usernum, stringer_t * name, uint64_t parent, uint32_t order, uint_t type)`

Insert a new folder into the database.

Parameters:

usernum the numerical id of the user to whom the new folder belongs.

name a managed string containing the name of the new folder.

parent the numerical id of the mail folder to be the parent of the new mail folder.

order the order number of this folder in its parent folder.

type the type of the new folder (can be `M_FOLDER_MESSAGES` or `M_FOLDER_CONTACTS`).

Returns:

0 on failure, or the numerical id of the newly inserted folder in the database on success.

Definition at line 91 of file `datatier.c`.

References `mm_wipe()`, `st_char_get()`, `st_length_get()`, and `stmt_insert()`.

Referenced by `contact_folder_create()`, and `message_folder_create()`.

5.106.1.4 `bool_t magma_folder_rename(uint64_t usernum, uint64_t foldernum, uint_t type, stringer_t * rename)`

Rename a folder in the database.

Parameters:

usernum the numerical id of the user requesting the folder renaming.

foldernum the numerical id of the folder to be renamed.

type the type of the folder to be renamed (can be `M_FOLDER_MESSAGES` or `M_FOLDER_CONTACTS`).

rename a managed string containing the new name of the specified folder.

Returns:

true on success or false on failure.

Definition at line 176 of file `datatier.c`.

References `mm_wipe()`, `st_char_get()`, `st_length_get()`, and `stmt_exec_affected()`.

Referenced by `contact_folder_rename()`.

5.107 src/objects/mail/datatier.c File Reference

```
#include "magma.h"
```

Functions

- void [mail_db_hide_message](#) (uint64_t messagenum)
Set a message invisible in the database.
- bool_t [mail_db_delete_message](#) (uint64_t usernum, uint64_t messagenum, uint32_t size, int_t transaction)
Delete a mail message from the mysql database and adjust the owner's quota.
- int_t [mail_db_update_message_folder](#) (uint64_t usernum, uint64_t messagenum, uint64_t source, uint64_t target, int64_t transaction)
Update a mail message's parent folder in the database.
- uint64_t [mail_db_insert_message](#) (uint64_t usernum, uint64_t foldernum, uint32_t status, uint32_t size, uint64_t signum, uint64_t sigkey, int_t transaction)
Insert a mail message into the database.
- uint64_t [mail_db_insert_duplicate_message](#) (uint64_t usernum, uint64_t foldernum, uint32_t status, uint32_t size, uint64_t signum, uint64_t sigkey, uint64_t created, int_t transaction)
Insert a duplicate entry for a message in the database.

5.107.1 Function Documentation

5.107.1.1 bool_t mail_db_delete_message (uint64_t usernum, uint64_t messagenum, uint32_t size, int_t transaction)

Delete a mail message from the mysql database and adjust the owner's quota. datatier.c

Parameters:

- usernum** the user id to whom the target mail message belongs.
- messagenum** the message id of the mail message to be deleted.
- size** the storage size, in bytes, of the message to be deleted.
- transaction** the mysql connection id on which to execute the statements.

Returns:

0 on failure or 1 on success.

Definition at line 41 of file datatier.c.

References [log_error](#), [mm_wipe\(\)](#), and [stmt_exec_affected_conn\(\)](#).

Referenced by [mail_remove_message\(\)](#).

5.107.1.2 void mail_db_hide_message (uint64_t messagenum)

Set a message invisible in the database.

Parameters:

- messagenum** the message id of the mail message to be hidden.

Returns:

This function returns no value.

Definition at line 15 of file `datatier.c`.

References `mm_wipe()`, and `stmt_exec()`.

Referenced by `mail_load_message()`.

5.107.1.3 `uint64_t mail_db_insert_duplicate_message (uint64_t usernum, uint64_t foldernum, uint32_t status, uint32_t size, uint64_t signum, uint64_t sigkey, uint64_t created, int_t transaction)`

Insert a duplicate entry for a message in the database.

Note:

This function will also update the user's storage quota information in the database.

Parameters:

usernum the numerical id of the user that owns the message.

foldernum the numerical id of the parent folder containing the message.

status the status flags value for the message.

size the size, in bytes, of the mail message on disk.

signum the spam signature for the message.

sigkey the spam key for the message.

created the UNIX timestamp of when the message was created.

transaction the transaction id for the database operation, in case the caller wants to roll back the transaction.

Returns:

NULL on failure, or the ID of the newly inserted message on success.

Definition at line 304 of file `datatier.c`.

References `magma_t::active`, `ISNULL`, `log_pedantic`, `magma`, `MAIL_MARK_JUNK`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `stmt_exec_conn()`, `stmt_insert_conn()`, and `magma_t::storage`.

Referenced by `mail_copy_message()`.

5.107.1.4 `uint64_t mail_db_insert_message (uint64_t usernum, uint64_t foldernum, uint32_t status, uint32_t size, uint64_t signum, uint64_t sigkey, int_t transaction)`

Insert a mail message into the database.

Note:

This function will also update the user's storage quota information in the database.

Parameters:

usernum the numerical id of the user to whom the mail message will belong.

foldernum the numerical id of the folder that will be the parent folder of the message.

status the status flags of the mail message.

size the size, in bytes, of the mail message on disk.

signum the spam signature for the message.

sigkey the spam key for the message.

transaction the transaction id for the database operation, in case the caller wants to roll back the transaction.

Returns:

0 on failure or the id of the newly inserted mail message on success.

Definition at line 172 of file datatier.c.

References magma_t::active, ISNULL, log_pedantic, magma, MAIL_MARK_JUNK, mm_wipe(), st_char_get(), st_length_get(), stmt_exec_conn(), stmt_insert_conn(), and magma_t::storage.

Referenced by mail_store_message().

5.107.1.5 **int_t mail_db_update_message_folder (uint64_t usernum, uint64_t messagenum, uint64_t source, uint64_t target, int64_t transaction)**

Update a mail message's parent folder in the database. usernum the numerical id of the user to whom the mail message belongs. messagenum the numerical id of the target mail message of the operation. source the numerical id of the parent folder in which the mail message currently resides. target the numerical id of the destination folder which is to be the new parent of the mail message. transaction a transaction id for the database operation, in case the caller needs to roll back changes on failure.

Returns:

-1 on failure, 0 if the target message could not be located in the database, or 1 on success.

Definition at line 100 of file datatier.c.

References log_pedantic, mm_wipe(), mysql_stmt_errno_d, mysql_stmt_error_d, pool_get_obj(), sql_pool, and stmt_exec_affected_conn().

Referenced by mail_move_message().

5.108 src/objects/messages/datatier.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t meta_data_fetch_folder_messages](#) (uint64_t usernum, [message_folder_t](#) *folder)

Populate a message folder with all of its child messages from the database.

- void [meta_data_fetch_message_tags](#) ([meta_message_t](#) *message)

Fetch the tags for a specified message from the database.

- [bool_t meta_data_fetch_messages](#) ([meta_user_t](#) *user)

Fetch all of a user's stored messages from the database and attach them to the meta user object.

5.108.1 Function Documentation

5.108.1.1 [bool_t meta_data_fetch_folder_messages](#) (uint64_t *usernum*, [message_folder_t](#) **folder*)

Populate a message folder with all of its child messages from the database. datatier.c

Parameters:

usernum the numerical id of the user that owns the folder.

folder a pointer to the message folder object to be populated.

Returns:

true on success or false on failure.

Definition at line 16 of file datatier.c.

References [inx_append\(\)](#), [log_error](#), [log_pedantic](#), [M_TYPE_UINT64](#), [message_alloc\(\)](#), [message_free\(\)](#), [message_t](#), [mm_wipe\(\)](#), [PLACER](#), [res_field_block\(\)](#), [res_field_length\(\)](#), [res_field_uint32\(\)](#), [res_field_uint64\(\)](#), [res_row_next\(\)](#), [res_table_free\(\)](#), [stmt_get_result\(\)](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [messages_update\(\)](#).

5.108.1.2 [void meta_data_fetch_message_tags](#) ([meta_message_t](#) **message*)

Fetch the tags for a specified message from the database.

Note:

The results of the operation will be stored in the specified meta message object's "tags" member.

Parameters:

message the meta message object for which the tags will be looked up.

Returns:

This function returns no value.

Definition at line 77 of file datatier.c.

References `ar_alloc()`, `ar_append()`, `ARRAY_TYPE_STRINGER`, `mm_wipe()`, `res_field_string()`, `res_row_count()`, `res_row_next()`, `res_table_free()`, and `stmt_get_result()`.

Referenced by `meta_data_fetch_messages()`, and `portal_endpoint_messages_tag()`.

5.108.1.3 `bool_t meta_data_fetch_messages (meta_user_t * user)`

Fetch all of a user's stored messages from the database and attach them to the meta user object.

Note:

Any of the user's existing messages will be destroyed first to allow for updates.

Parameters:

user the meta user object whose mail messages will be retrieved.

Returns:

true on success or false on failure.

TODO: Do we still need this once the refactorization is complete?

Definition at line 115 of file datatier.c.

References `inx_alloc()`, `inx_append()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_truncate()`, `log_error`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `MAIL_STATUS_TAGGED`, `meta_data_fetch_message_tags()`, `meta_message_free()`, `meta_message_t`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `mm_wipe()`, `res_field_block()`, `res_field_length()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::type`, `multi_t::u64`, and `multi_t::val`.

Referenced by `meta_messages_login_update()`, and `meta_messages_update()`.

5.109 src/objects/meta/datatier.c File Reference

```
#include "magma.h"
```

Functions

- void [meta_data_update_log](#) (meta_user_t *user, META_PROTOCOL prot)
Update the per-user entry in the Log table for the specified protocol.
- bool_t [meta_data_fetch_folders](#) (meta_user_t *user)
Retrieve all of a user's message folders from the database.
- int_t [meta_data_fetch_mailbox_aliases](#) (meta_user_t *user)
Get all the mailbox aliases for a specified user.
- int_t [meta_data_fetch_user](#) (meta_user_t *user)
Build a meta user object by username, hashed password, and hashed key storage password.
- int_t [meta_data_fetch_keys](#) (meta_user_t *user, key_pair_t *output, int64_t transaction)
Retrieve the encryption keys for a user account.
- int_t [meta_data_insert_keys](#) (uint64_t usernum, stringer_t *username, key_pair_t *input, int64_t transaction)
Store the encryption keys for a user account.
- int_t [meta_data_fetch_shard](#) (uint64_t usernum, uint16_t serial, stringer_t *label, stringer_t *output, uint_t *rotated, int64_t transaction)
Retrieve a shard value for user account.
- int_t [meta_data_insert_shard](#) (uint64_t usernum, uint16_t serial, stringer_t *label, stringer_t *shard, int64_t transaction)
Store the user realm shard value.
- bool_t [meta_data_acknowledge_alert](#) (uint64_t alertnum, uint64_t usernum, uint32_t transaction)
Mark a user alert message as acknowledged in the database.
- inx_t * [meta_data_fetch_alerts](#) (uint64_t usernum)
Get all unacknowledged alert messages for a user.
- bool_t [meta_data_flags_replace](#) (inx_t *messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)
Remove all user (non-system) flags from a collection of mail messages, and set the specified flags mask for them.
- bool_t [meta_data_flags_remove](#) (inx_t *messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)
Remove the specified flags mask from a collection of mail messages.
- bool_t [meta_data_flags_add](#) (inx_t *messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)
Add the specified flags mask to a collection of mail messages.
- uint64_t [meta_data_delete_folder](#) (uint64_t usernum, uint64_t foldernum)
Delete a message folder from the database.
- uint64_t [meta_data_update_folder_name](#) (uint64_t usernum, uint64_t foldernum, stringer_t *name, uint64_t parent, uint32_t order)
Update the record for a message folder in the database.

- `uint64_t meta_data_insert_folder` (`uint64_t usernum`, `stringer_t *name`, `uint64_t parent`, `uint32_t order`)
Insert a new mail folder into the database.
- `int_t meta_data_insert_tag` (`meta_message_t *message`, `stringer_t *tag`)
Insert a tag for a message into the database.
- `int_t meta_data_truncate_tags` (`meta_message_t *message`)
Remove all tags associated with a message in the database.
- `int_t meta_data_delete_tag` (`meta_message_t *message`, `stringer_t *tag`)
Remove a tag from a message in the database.
- `inx_t * meta_data_fetch_all_tags` (`uint64_t usernum`)
Fetch all the tags attached to messages of a specified user from the database.

5.109.1 Function Documentation

5.109.1.1 `bool_t meta_data_acknowledge_alert` (`uint64_t alertnum`, `uint64_t usernum`, `uint32_t transaction`)

Mark a user alert message as acknowledged in the database. `datatier.c`

Note:

If the table is not updated immediately, another check is made to see if the alert is still pending. If so, false is returned.

Parameters:

alertnum the numerical id of the alert message to be acknowledged.

usernum the numerical id of the user to whom the alert message belongs.

transaction the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

true if the alert was acknowledged successfully, or false on failure.

Definition at line 584 of file `datatier.c`.

References `log_pedantic`, `mm_wipe()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_exec_affected_conn()`, and `stmt_get_result_conn()`.

Referenced by `portal_endpoint_alert_acknowledge()`.

5.109.1.2 `uint64_t meta_data_delete_folder` (`uint64_t usernum`, `uint64_t foldernum`)

Delete a message folder from the database.

Parameters:

usernum the numerical id of the user that owns the folder to be deleted.

foldernum the folder id of the message folder to be deleted.

Returns:

1 if the message folder was successfully, or ≤ 0 if there was an error.

Definition at line 932 of file `datatier.c`.

References `M_FOLDER_MESSAGES`, `mm_wipe()`, `stmt_exec_affected()`, and `type()`.

Referenced by `imap_folder_remove()`.

5.109.1.3 `int_t meta_data_delete_tag (meta_message_t * message, stringer_t * tag)`

Remove a tag from a message in the database.

Parameters:

message a pointer to the meta message object of the message to have the tag stripped.

tag a managed string containing the name of the tag to be deleted.

Returns:

0 on success or -1 on failure.

Definition at line 1137 of file `datatier.c`.

References `mm_wipe()`, `st_char_get()`, `st_length_get()`, and `stmt_exec_affected()`.

Referenced by `portal_endpoint_messages_tag()`.

5.109.1.4 `inx_t* meta_data_fetch_alerts (uint64_t usernum)`

Get all unacknowledged alert messages for a user.

Parameters:

usernum the numerical id of the user for whom the alert messages will be fetched.

Returns:

NULL on failure, or a pointer to an `inx` holder containing all of the user's alert messages on success.

Definition at line 649 of file `datatier.c`.

References `alert_alloc()`, `inx_alloc()`, `inx_free()`, `inx_insert()`, `inx_t`, `log_error`, `log_info`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_alert_t`, `mm_free()`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_endpoint_alert_list()`.

5.109.1.5 `inx_t* meta_data_fetch_all_tags (uint64_t usernum)`

Fetch all the tags attached to messages of a specified user from the database.

Parameters:

usernum the numerical id of the user for whom the message tags will be fetched.

Returns:

NULL on failure, or an `inx` holder containing a list of all the user's messages' tags as managed strings on success.

Definition at line 1170 of file `datatier.c`.

References `inx_alloc()`, `inx_free()`, `inx_insert()`, `inx_t`, `log_error`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `res_field_string()`, `res_row_next()`, `res_table_free()`, `st_free()`, `stmt_get_result()`, `multi_t::type`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_endpoint_messages_tags()`.

5.109.1.6 bool_t meta_data_fetch_folders (meta_user_t * user)

Retrieve all of a user's message folders from the database.

Note:

If the user already had a working set of message folders they will be deleted first.

Parameters:

user a pointer to the meta user object of the user making the request, which will be updated on success.

Returns:

-1 on failure or 1 on success.

Definition at line 58 of file datatier.c.

References `inx_alloc()`, `inx_cleanup()`, `inx_insert()`, `log_error`, `log_pedantic`, `M_FOLDER_MESSAGES`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_folder_t`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `mm_wipe()`, `res_field_block()`, `res_field_length()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::type`, `type()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `meta_update_folders()`.

5.109.1.7 int_t meta_data_fetch_keys (meta_user_t * user, key_pair_t * output, int64_t transaction)

Retrieve the encryption keys for a user account.

Parameters:

user a meta object with the user number field populated.

output a key pair object to hold the results.

transaction the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

-1 for unexpected program/system error, 0 on success, 1 if no rows are found.

Definition at line 307 of file datatier.c.

References `base64_decode_mod()`, `log_pedantic`, `mm_wipe()`, `PLACER`, `key_pair_t::private`, `key_pair_t::public`, `res_field_block()`, `res_field_length()`, `res_row_next()`, `res_table_free()`, `st_char_get()`, `st_empty`, `st_free()`, `st_length_int()`, and `stmt_get_result_conn()`.

Referenced by `meta_update_keys()`.

5.109.1.8 int_t meta_data_fetch_mailbox_aliases (meta_user_t * user)

Get all the mailbox aliases for a specified user.

Parameters:

user a pointer to the partially populated meta user object to be queried, with its `usenum` field set.

Returns:

true on success or false on failure.

Definition at line 159 of file datatier.c.

References `alias_alloc()`, `inx_alloc()`, `inx_cleanup()`, `inx_insert()`, `log_error`, `log_info`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_alias_t`, `mm_free()`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint64()`, `res_field_uint8()`, `res_row_next()`, `res_table_free()`, `st_char_get()`, `st_length_int()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `meta_update_aliases()`.

5.109.1.9 `int_t meta_data_fetch_shard (uint64_t usernum, uint16_t serial, stringer_t * label, stringer_t * output, uint_t * rotated, int64_t transaction)`

Retrieve a shard value for user account.

Parameters:

- usernum* the numerical id of the user to whom the alert message belongs.
- serial* the serial number associated with the shard value.
- label* the textual label associated with the shard value.
- output* the buffer where the binary output should be stored.
- rotated* the buffer where the rotated variable is stored.
- transaction* the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

- 1 for unexpected program/system error, 0 on success, 1 if no rows are found.

Definition at line 440 of file `datatier.c`.

References `base64_decode_mod()`, `log_pedantic`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint8()`, `res_row_next()`, `res_table_free()`, `st_avail_get()`, `st_char_get()`, `st_data_get()`, `st_empty`, `st_length_get()`, `st_length_int()`, `st_opt_get()`, `st_valid_destination()`, `STACIE_SHARD_LENGTH`, and `stmt_get_result_conn()`.

Referenced by `meta_update_realms()`.

5.109.1.10 `int_t meta_data_fetch_user (meta_user_t * user)`

Build a meta user object by username, hashed password, and hashed key storage password.

Parameters:

- user* a meta object with the user number field populated.

Returns:

- 1 for unexpected program/system error, 0 on success, 1 if no rows are found.

Definition at line 224 of file `datatier.c`.

References `base64_decode_mod()`, `log_pedantic`, `META_USER_ENCRYPT_DATA`, `META_USER_OVERQUOTA`, `META_USER_TLS`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_int8()`, `res_field_length()`, `res_field_string()`, `res_row_next()`, `res_table_free()`, `st_cleanup`, `st_empty`, and `stmt_get_result()`.

Referenced by `meta_update_user()`.

5.109.1.11 `bool_t meta_data_flags_add (inx_t * messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)`

Add the specified flags mask to a collection of mail messages.

Parameters:

- messages* an `inx` holder containing the collection of messages to have their flags updated.
- usernum* the numerical id of the user to whom the target messages belong, for validation purposes.
- foldernum* the numerical id of the parent folder containing the messages to be updated, for validation purposes.
- flags* a mask of all flags that are to be added to any matching messages in the collection.

Returns:

true on success or false on failure.

Definition at line 869 of file datatier.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `meta_message_t`, `mm_wipe()`, and `stmt_exec()`.

Referenced by `imap_fetch()`, `imap_update_flags()`, `portal_endpoint_messages_flag()`, and `portal_endpoint_messages_tag()`.

5.109.1.12 `bool_t meta_data_flags_remove (inx_t * messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)`

Remove the specified flags mask from a collection of mail messages.

Parameters:

messages an inx holder containing the collection of messages to have their flags removed.

usernum the numerical id of the user to whom the target messages belong, for validation purposes.

foldernum the numerical id of the parent folder containing the messages to be updated, for validation purposes.

flags a mask of all flags that are to be stripped from any matching messages in the collection.

Returns:

true on success or false on failure.

Definition at line 795 of file datatier.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `meta_message_t`, `mm_wipe()`, and `stmt_exec()`.

Referenced by `imap_select()`, `imap_update_flags()`, `portal_endpoint_messages_flag()`, and `portal_endpoint_messages_tag()`.

5.109.1.13 `bool_t meta_data_flags_replace (inx_t * messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)`

Remove all user (non-system) flags from a collection of mail messages, and set the specified flags mask for them.

Note:

The new mask can contain both user and system flags, but only user flags will be stripped from each message initially.

Parameters:

messages an inx holder containing the collection of messages to have their flags updated.

usernum the numerical of the user to whom the target messages belong, for validation purposes.

foldernum the numerical id of the parent folder containing the messages to be updated, for validation purposes.

flags a mask of all flags that are to be added to any matching messages in the collection.

Returns:

true on success or false on failure.

Definition at line 713 of file datatier.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `MAIL_STATUS_USER_FLAGS`, `meta_message_t`, `mm_wipe()`, and `stmt_exec()`.

Referenced by `imap_update_flags()`, and `portal_endpoint_messages_flag()`.

5.109.1.14 uint64_t meta_data_insert_folder (uint64_t usernum, stringer_t * name, uint64_t parent, uint32_t order)

Insert a new mail folder into the database.

Parameters:

- usernum* the numerical id of the user to whom the new folder belongs.
- name* a managed string containing the name of the new mail folder.
- parent* the numerical id of the mail folder to be the parent of the new mail folder.
- order* the order number of this folder in its parent folder.

Returns:

0 on failure, or the numerical id of the newly inserted mail folder in the database on success.

Definition at line 1028 of file datatier.c.

References M_FOLDER_MESSAGES, mm_wipe(), st_char_get(), st_length_get(), stmt_insert(), and type().

Referenced by imap_folder_create(), and imap_folder_rename().

5.109.1.15 int_t meta_data_insert_keys (uint64_t usernum, stringer_t * username, key_pair_t * input, int64_t transaction)

Store the encryption keys for a user account.

Parameters:

- usernum* the numerical id of the user to whom the alert message belongs.
- username* the plain text username.
- input* the public and private key pair.
- transaction* the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

-1 for unexpected program/system error, 0 on success, 1 if the query executes, but no rows are affected.

Definition at line 372 of file datatier.c.

References base64_encode_mod(), log_pedantic, mm_wipe(), key_pair_t::private, key_pair_t::public, st_char_get(), st_cleanup, st_length_get(), st_length_int(), st_populated, and stmt_exec_affected_conn().

Referenced by meta_crypto_keys_create().

5.109.1.16 int_t meta_data_insert_shard (uint64_t usernum, uint16_t serial, stringer_t * label, stringer_t * shard, int64_t transaction)

Store the user realm shard value.

Parameters:

- usernum* the numerical id of the user to whom the alert message belongs.
- serial* the nummerid serial number associated with the shard value.
- label* the textual label associated with the shard value.
- shard* the binary shard value.
- transaction* the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

-1 for unexpected program/system error, 0 on success, 1 if the query executes, but no rows are affected.

Definition at line 508 of file datatier.c.

References `base64_encode_mod()`, `log_pedantic`, `MANAGEDBUF`, `mm_wipe()`, `st_char_get()`, `st_data_get()`, `st_empty`, `st_length_get()`, `st_length_int()`, `STACIE_SHARD_LENGTH`, and `stmt_exec_affected_conn()`.

Referenced by `meta_update_realms()`.

5.109.1.17 int_t meta_data_insert_tag (meta_message_t * message, stringer_t * tag)

Insert a tag for a message into the database.

Parameters:

message the meta message object of the message to be tagged.

tag a managed string containing the name of the tag.

Returns:

0 on success or -1 on failure.

Definition at line 1075 of file datatier.c.

References `mm_wipe()`, `st_char_get()`, `st_length_get()`, and `stmt_exec_affected()`.

Referenced by `portal_endpoint_messages_tag()`.

5.109.1.18 int_t meta_data_truncate_tags (meta_message_t * message)

Remove all tags associated with a message in the database.

Parameters:

message a pointer to the meta message object of the message to have all of its tags stripped.

Returns:

0 on success or -1 on failure.

Definition at line 1108 of file datatier.c.

References `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `portal_endpoint_messages_tag()`.

5.109.1.19 uint64_t meta_data_update_folder_name (uint64_t usernum, uint64_t foldernum, stringer_t * name, uint64_t parent, uint32_t order)

Update the record for a message folder in the database.

Parameters:

usernum the numerical id of the user that owns the specified folder.

foldernum the id of the folder to have its properties adjusted.

name a managed string containing the new name of the specified folder.

parent the id of the new parent folder to be set for the specified message folder.

order the value of the order for the specified folder.

Returns:

1 on success, or ≤ 0 on failure.

Definition at line 972 of file `datatier.c`.

References `M_FOLDER_MESSAGES`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `stmt_exec_affected()`, and `type()`.

Referenced by `imap_folder_rename()`.

5.109.1.20 void meta_data_update_log (meta_user_t * *user*, META_PROTOCOL *prot*)

Update the per-user entry in the Log table for the specified protocol.

Note:

This function will set the last session timestamp for the user, and increment the sessions counter in the database.

Parameters:

user the meta user object of the user making the logging request.

prot the protocol associated with the log request: `META_PROT_POP`, `META_PROT_IMAP`, or `META_PROT_WEB`.

Returns:

This function returns no value.

Definition at line 20 of file `datatier.c`.

References `log_pedantic`, `META_PROTOCOL_IMAP`, `META_PROTOCOL_POP`, `META_PROTOCOL_WEB`, `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `imap_login()`, and `pop_pass()`.

5.110 src/objects/warehouse/datatier.c File Reference

```
#include "magma.h"
```

Functions

- [inx_t * warehouse_fetch_domains](#) (void)
Fetch the list of configured domains from the database.
- [inx_t * warehouse_fetch_patterns](#) (void)
Fetch the pattern list from the database.

5.110.1 Function Documentation

5.110.1.1 inx_t* warehouse_fetch_domains (void)

Fetch the list of configured domains from the database. datatier.c

Returns:

NULL on failure, or an inx object holding all the configured domains on success.

Definition at line 14 of file datatier.c.

References [domain_alloc\(\)](#), [domain_t](#), [inx_alloc\(\)](#), [inx_free\(\)](#), [inx_insert\(\)](#), [inx_t](#), [log_check](#), [log_info](#), [log_pedantic](#), [M_INX_TREE](#), [M_TYPE_STRINGER](#), [MAGMA_HOSTNAME_MAX](#), [mm_free\(\)](#), [PLACER](#), [res_field_block\(\)](#), [res_field_int8\(\)](#), [res_field_length\(\)](#), [res_row_next\(\)](#), [res_table_free\(\)](#), [multi_t::st](#), [stmt_get_result\(\)](#), and [multi_t::val](#).

Referenced by [domain_start\(\)](#).

5.110.1.2 inx_t* warehouse_fetch_patterns (void)

Fetch the pattern list from the database.

Returns:

NULL on failure or a pointer to an inx holder containing a collection of managed strings with the patterns on success.

Definition at line 64 of file datatier.c.

References [inx_alloc\(\)](#), [inx_free\(\)](#), [inx_insert\(\)](#), [inx_t](#), [log_pedantic](#), [M_INX_LINKED](#), [M_TYPE_UINT64](#), [res_field_string\(\)](#), [res_row_next\(\)](#), [res_table_free\(\)](#), [st_free\(\)](#), [stmt_get_result\(\)](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [pattern_update\(\)](#).

5.111 src/servers/smtp/datatier.c File Reference

```
#include "magma.h"
```

Functions

- `int_t smtp_get_action (chr_t *string, size_t length)`
Parse an smtp event action string from the Dispatch table.
- `stringer_t * smtp_fetch_autoreply (uint64_t autoreply, uint64_t usernum)`
Fetch a specified auto-reply message for a user.
- `int_t smtp_fetch_inbound (stringer_t *address, smtp_inbound_prefs_t **output)`
Fetch a user's SMTP preferences for inbound mail.
- `table_t * smtp_fetch_rolmessages (uint64_t usernum)`
Retrieve a list, at a maximum of 20 entries, of the oldest messages owned by a user.
- `void smtp_update_receive_stats (connection_t *con, smtp_inbound_prefs_t *prefs)`
Update the receiving statistics and per-user log tables in the database for a successfully received smtp message.
- `uint64_t smtp_insert_spamsig (smtp_inbound_prefs_t *prefs, uint64_t key, int_t code)`
Store a spam signature in the database.
- `void smtp_update_transmission_stats (connection_t *con)`
Update the transmission and per-user log tables in the database for a successfully sent smtp message.
- `int_t smtp_check_transmit_quota (uint64_t usernum, size_t num_recipients, smtp_outbound_prefs_t *prefs)`
Check to see if a user's current mail send request would push them over their daily transmission quota.
- `int_t smtp_fetch_authorization (stringer_t *username, stringer_t *verification, smtp_outbound_prefs_t **output)`
Check if a user is authorized to send messages, and retrieve the user's outbound smtp preferences.
- `int_t smtp_check_receive_quota (connection_t *con, smtp_inbound_prefs_t *prefs)`
Check the user's received statistics from the database to see if receiving a message would result in a quota overage.
- `int_t smtp_check_authorized_from (uint64_t usernum, stringer_t *address)`
Check to see if a user is permitted to send email originating from a specified email address.

5.111.1 Function Documentation

5.111.1.1 int_t smtp_check_authorized_from (uint64_t usernum, stringer_t * address)

Check to see if a user is permitted to send email originating from a specified email address. datatier.c

Note:

The authorization attempt first checks against the specified email address, and if unsuccessful, upon the domain component of the email address if wildcards are enabled for that domain.

Parameters:

usenum the numerical id of the user attempting to send the mail message.

address a pointer to a managed string containing the From address value of the mail message to be sent.

Returns:

1 if the user is authorized to send email from the specified address, or 0 otherwise.

Definition at line 761 of file datatier.c.

References domain_wildcard(), mail_domain_get(), mm_wipe(), placer_t, res_row_count(), res_table_free(), st_char_get(), st_length_get(), and stmt_get_result().

Referenced by portal_outbound_checks(), and smtp_data_outbound().

5.111.1.2 int_t smtp_check_receive_quota (connection_t * con, smtp_inbound_prefs_t * prefs)

Check the user's received statistics from the database to see if receiving a message would result in a quota overage.

Note:

The sum total of all emails received by the user over the past 24 hours is calculated from the database, and these checks are made: 1. The amount of mail messages received in the past 24 hours by the user does not exceed their daily mail received quota. 2. The messages received in the past 24 hours by the user from this subnet does not exceed the user's daily per-subnet received quota.

Parameters:

con a pointer to the connection object over which the smtp message was received.

prefs a pointer to the user's smtp inbound mail preferences.

Returns:

0 if message receipt is permitted, 1 if the general daily receiving limit was exceeded, 2 if the sending subnet's transmission limit was exceeded, or -1 if there was a general error.

Definition at line 696 of file datatier.c.

References con_addr_subnet(), smtp_inbound_prefs_t::daily_rcv_limit, smtp_inbound_prefs_t::daily_rcv_limit_ip, log_pedantic, mm_wipe(), number, res_field_block(), res_field_length(), res_field_uint64(), res_row_next(), res_table_free(), st_char_get(), st_free(), st_length_get(), stmt_get_result(), uint64_conv_bl(), and smtp_inbound_prefs_t::usernum.

Referenced by smtp_rcpt_to().

5.111.1.3 int_t smtp_check_transmit_quota (uint64_t usernum, size_t num_recipients, smtp_outbound_prefs_t * prefs)

Check to see if a user's current mail send request would push them over their daily transmission quota.

Note:

This check is performed by querying the database to see how many messages a user has sent in the past 24 hour period, and by adding the current number of recipients of the pending email request to that number to see if their quota would be exceeded.

Parameters:

con a pointer to the connection object of the user attempting to send mail.

Returns:

-1 on error, 0 if the send operation is permitted, or 1 if the send operation would result in a daily send quota overage.

Definition at line 543 of file datatier.c.

References smtp_outbound_prefs_t::daily_send_limit, log_pedantic, mm_wipe(), res_field_uint64(), res_row_next(), res_table_free(), smtp_outbound_prefs_t::sent_today, and stmt_get_result().

Referenced by portal_outbound_checks(), and smtp_data_outbound().

5.111.1.4 `int_t smtp_fetch_authorization (stringer_t * username, stringer_t * verification, smtp_outbound_prefs_t ** output)`

Check if a user is authorized to send messages, and retrieve the user's outbound smtp preferences.

Note:

This function first checks if the account is locked in the Users table; next it populates the user's outbound mail preferences with a combination of data from the Users and Dispatch tables. The number of messages the user has sent in the past 24 hours is also computed from the Transmitting table.

Parameters:

cred a pointer to a credential object for the user, which must be of type CREDENTIAL_AUTH.
output a pointer to the address of an outbound smtp preferences object to receive the value of the lookup.

Returns:

0 on success or < 0 for failures, and > 0 for account locks. -2: general server or database error. -1: authentication failure (invalid username/password combination). 0: successful authentication

Definition at line 594 of file `datatier.c`.

References `base64_encode_mod()`, `smtp_outbound_prefs_t::daily_send_limit`, `smtp_outbound_prefs_t::domain`, `smtp_outbound_prefs_t::importance`, `log_error`, `log_pedantic`, `mm_alloc()`, `mm_free()`, `mm_wipe()`, `res_field_int8()`, `res_field_string()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `smtp_outbound_prefs_t::send_size_limit`, `smtp_outbound_prefs_t::sent_today`, `st_char_get()`, `st_free()`, `st_length_get()`, `st_length_int()`, `st_populated`, `stmt_get_result()`, `smtp_outbound_prefs_t::tls`, and `smtp_outbound_prefs_t::usernum`.

Referenced by `portal_outbound_checks()`, `smtp_auth_login()`, and `smtp_auth_plain()`.

5.111.1.5 `stringer_t* smtp_fetch_autoreply (uint64_t autoreply, uint64_t usernum)`

Fetch a specified auto-reply message for a user.

Note:

This function first checks the cache, and falls back to the database.

Parameters:

autoreply the numerical id of the auto-reply message in the database.
usernum the numerical id of the user to whom the auto-reply message belongs.

Returns:

NULL on failure, or a pointer to a managed string containing the user's auto-reply on success.

Definition at line 50 of file `datatier.c`.

References `cache_get()`, `cache_set()`, `log_pedantic`, `mm_wipe()`, `PLACER`, `res_field_string()`, `res_row_next()`, `res_table_free()`, and `stmt_get_result()`.

Referenced by `smtp_reply()`.

5.111.1.6 `int_t smtp_fetch_inbound (stringer_t * address, smtp_inbound_prefs_t ** output)`

Fetch a user's SMTP preferences for inbound mail.

Parameters:

cred a pointer to the credential object of a user with

address

Returns:

0 for success or < 0 for failures, and > 0 for account locks.

Return values:

-3,: for general server and database errors.

-2,: if the domain isn't local.

-1,: if the address is local but we didn't find a matching mailbox.

0,: everything worked

1,: the account is locked for inactivity (invalid username/password combination).

See also:

[AUTH_LOCK_INACTIVITY](#)

Return values:

2,: the account plan has expired.

See also:

[AUTH_LOCK_EXPIRED](#)

Return values:

3,: the account is subject to an administrative lock.

See also:

[AUTH_LOCK_ADMIN](#)

Return values:

4,: the account is locked due to suspicion of abuse violations.

See also:

[AUTH_LOCK_ABUSE](#)

Return values:

5,: the account has been locked at the request of the user.

See also:

[AUTH_LOCK_USER](#)

Definition at line 119 of file datatier.c.

References `smtp_inbound_filter_t::action`, `smtp_inbound_prefs_t::address`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_NONE`, `smtp_inbound_prefs_t::autoreply`, `base64_decode_mod()`, `BINARY`, `smtp_inbound_prefs_t::bounces`, `CONTIGUOUS`, `smtp_inbound_prefs_t::daily_recv_limit`, `smtp_inbound_prefs_t::daily_recv_limit_ip`, `smtp_inbound_prefs_t::dkim`, `smtp_inbound_prefs_t::dkimaction`, `smtp_inbound_prefs_t::domain`, `domain_wildcard()`, `smtp_inbound_filter_t::expression`, `smtp_inbound_filter_t::field`, `smtp_inbound_prefs_t::filters`, `smtp_inbound_filter_t::foldernum`, `smtp_inbound_prefs_t::forwarded`, `smtp_inbound_prefs_t::greylist`, `smtp_inbound_prefs_t::greytime`, `HEAP`, `smtp_inbound_prefs_t::inbox`, `inx_alloc()`, `inx_insert()`, `smtp_inbound_filter_t::label`, `smtp_inbound_filter_t::location`, `log_error`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `mail_domain_get()`, `MANAGED_T`, `mm_alloc()`, `mm_free()`, `mm_wipe()`, `NONE`, `smtp_inbound_prefs_t::overquota`, `smtp_inbound_prefs_t::phish`, `smtp_inbound_prefs_t::phishaction`, `PLACER`, `placer_t`, `prime_set()`, `smtp_inbound_prefs_t::quota`, `smtp_inbound_prefs_t::rbl`, `smtp_inbound_prefs_t::rblaction`, `smtp_inbound_prefs_t::rcptto`,

smtp_inbound_prefs_t::recv_size_limit, res_field_block(), res_field_int8(), res_field_length(), res_field_string(), res_field_uint32(), res_field_uint64(), res_row_count(), res_row_next(), res_table_free(), smtp_inbound_prefs_t::rollout, smtp_inbound_filter_t::rulenum, smtp_inbound_prefs_t::secure, smtp_inbound_prefs_t::signet, SMTP_FILTER_ACTION_LABEL, SMTP_FILTER_ACTION_MOVE, SMTP_FILTER_LOCATION_FIELD, smtp_free_inbound(), smtp_get_action(), smtp_list_free_filter(), smtp_inbound_prefs_t::spam, smtp_inbound_prefs_t::spamaction, smtp_inbound_prefs_t::spf, smtp_inbound_prefs_t::spfaction, st_char_get(), st_cleanup, st_dupe(), st_dupe_opts(), st_empty, st_length_get(), st_length_int(), stmt_get_result(), smtp_inbound_prefs_t::stor_size, smtp_inbound_filter_t::type, multi_t::u64, smtp_inbound_prefs_t::usernum, multi_t::val, smtp_inbound_prefs_t::virus, and smtp_inbound_prefs_t::virusaction.

Referenced by smtp_rcpt_to().

5.111.1.7 table_t* smtp_fetch_rollmessages (uint64_t usernum)

Retrieve a list, at a maximum of 20 entries, of the oldest messages owned by a user.

Parameters:

usernum the numerical id of the user whose messages are to be queried.

Returns:

NULL on failure, or a pointer to a sql results set containing the user's oldest messages on success.

Definition at line 345 of file datatier.c.

References log_pedantic, mm_wipe(), and stmt_get_result().

Referenced by smtp_rollout().

5.111.1.8 int_t smtp_get_action (chr_t * string, size_t length)

Parse an smtp event action string from the Dispatch table.

Note:

These values describe user-specified actions for events related to the spam filter, virus scanner, phishing detection, SPF, DKIM, and RBL checks.

Parameters:

string a pointer to a null-terminated string containing the name of the action.

length the length, in bytes, of the action string.

Returns:

the code of the corresponding smtp event action, SMTP_ACTION_UNDEFINED if the action is not known, or SMTP_ACTION_ERROR on failure.

Definition at line 18 of file datatier.c.

References log_error, PLACER, SMTP_ACTION_BOUNCE, SMTP_ACTION_DELETE, SMTP_ACTION_ERROR, SMTP_ACTION_MARK, SMTP_ACTION_MARK_READ, SMTP_ACTION_REJECT, SMTP_ACTION_UNDEFINED, and st_cmp_cs_eq().

Referenced by smtp_fetch_inbound().

5.111.1.9 uint64_t smtp_insert_spamsig (smtp_inbound_prefs_t * prefs, uint64_t key, int_t code)

Store a spam signature in the database.

See also:

dspam_process()

Parameters:

prefs a pointer to the specified user's inbound mail preferences data.
key a randomly chosen authentication key for the signature.
code the dspam return code for the message from dspam_process()

Returns:

0 on failure, or the number of the newly inserted spam signature on success.

Definition at line 446 of file datatier.c.

References log_pedantic, mm_wipe(), smtp_inbound_prefs_t::spamsig, st_char_get(), st_length_get(), stmt_insert(), and smtp_inbound_prefs_t::usernum.

Referenced by smtp_store_spamsig().

5.111.1.10 void smtp_update_receive_stats (connection_t * con, smtp_inbound_prefs_t * prefs)

Update the receiving statistics and per-user log tables in the database for a successfully received smtp message.

Note:

The Receiving table is updated with the subnet address from which the message was received; the Log table for the user is updated to reflect the newly calculated totals of bounces or messages received.

Parameters:

con the connection across which the smtp message was received.
prefs a pointer to the user's smtp inbound mail preferences.

Returns:

This function returns no value.

Definition at line 374 of file datatier.c.

References smtp_inbound_prefs_t::bounces, con_addr_subnet(), log_pedantic, mm_wipe(), PLACER, st_char_get(), st_cmp_cs_eq(), st_free(), st_length_get(), stmt_exec(), and smtp_inbound_prefs_t::usernum.

Referenced by smtp_accept_message().

5.111.1.11 void smtp_update_transmission_stats (connection_t * con)

Update the transmission and per-user log tables in the database for a successfully sent smtp message.

Note:

The Transmitting table is updated with the timestamp of this transaction; the Log table for the user is updated to reflect the newly calculated total for messages sent.

Parameters:

con a pointer to the connection object across which the smtp message was sent.
prefs a pointer to the user's smtp inbound mail preferences.

Returns:

This function returns no value.

Definition at line 498 of file `datatier.c`.

References `log_pedantic`, `mm_wipe()`, and `stmt_exec()`.

Referenced by `smtp_data_outbound()`.

5.112 src/web/register/datatier.c File Reference

```
#include "magma.h"
```

Functions

- `inx_t * register_data_fetch_blocklist` (void)
Fetch the blocklist for new user registration from the database.
- `bool_t register_data_check_username` (stringer_t *username)
Check to see if a username requested by a registration attempt has already been taken.
- `int_t register_data_insert_user` (connection_t *con, uint16_t plan, stringer_t *username, stringer_t *password, int64_t transaction, uint64_t *outuser)
Insert a newly registered user into the database using information gathered by registration step #2.

5.112.1 Function Documentation

5.112.1.1 bool_t register_data_check_username (stringer_t * username)

Check to see if a username requested by a registration attempt has already been taken. datatier.c

Parameters:

username the username to be checked against the database.

Returns:

true if the username is taken or false if it is not.

Definition at line 67 of file datatier.c.

References `mm_wipe()`, `res_row_next()`, `res_table_free()`, `st_char_get()`, `st_length_get()`, and `stmt_get_result()`.

Referenced by `register_business_step1()`.

5.112.1.2 inx_t* register_data_fetch_blocklist (void)

Fetch the blocklist for new user registration from the database. HIGH: The prepared statements being used aren't valid. The queries need to be copied over and created. `register_fetch_blocklist` still needs to be defined.

Returns:

an inx holder containing the registration blocklist as a collection of managed strings.

Definition at line 18 of file datatier.c.

References `inx_alloc()`, `inx_free()`, `inx_insert()`, `inx_t`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `res_field_string()`, `res_row_next()`, `res_table_free()`, `st_free()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `register_blocklist_update()`.

5.112.1.3 `int_t register_data_insert_user(connection_t * con, uint16_t plan, stringer_t * username, stringer_t * password, int64_t transaction, uint64_t * outuser)`

Insert a newly registered user into the database using information gathered by registration step #2.

Note:

The following steps occur: 1. Insert a new user into the Users table, supplying username, hashed password, plan info, and quota. 2. Insert a random shard value into the User_Realms table. 3. Insert a blank entry into the Profile table for the user. 4. Insert an entry into the Folders table for the user's "Inbox" folder. 5. Insert a new entry into the Log table containing the usernum and IP address of the client request. 6. Insert a new entry into the Dispatch table for the user, configuring the spam folder, inbox, send/receive/daily send/daily receive limits, etc. 7. Insert a new entry into the Mailboxes table for the user.

Parameters:

con a pointer to the connection object of the client making the registration request.

reg the current registration session of the user to be added.

transaction a mysql transaction id for all database operations, since they all need to be committed atomically or rolled back.

outuser a pointer to a numerical id to receive the newly generated and inserted user id.

Returns:

0 if the new user account was successfully created, -1 if a technical error occurs, and 1 if an invalid value is provided.

LOW: This function should be passed a number of days (or years) to use for any of the pre-paid account plans. LOW: The IP address could be passed in as an `ip_t` or as a string to avoid the need for the entire connection object.

Definition at line 114 of file `datatier.c`.

References `auth_stacie()`, `auth_stacie_cleanup()`, `base64_encode_mod()`, `con_addr_presentation()`, `magma_t::domain`, `auth_stacie_t::keys`, `log_pedantic`, `magma`, `MANAGEDBUF`, `auth_stacie_t::master`, `meta_crypto_keys_create()`, `magma_t::minimum_password_length`, `mm_wipe()`, `ns_length_get()`, `PLACER`, `magma_t::secure`, `st_char_get()`, `st_cleanup`, `st_data_get()`, `st_length_get()`, `st_length_int()`, `st_merge`, `st_search_chr()`, `st_sprint()`, `stacie_create_salt()`, `stacie_create_shard()`, `stacie_realm_key()`, `stmt_exec_conn()`, `stmt_insert_conn()`, `magma_t::system`, `time_print_local()`, `auth_stacie_t::tokens`, `utf8_length_st()`, and `auth_stacie_t::verification`.

Referenced by `api_endpoint_register()`, and `register_business_step2()`.

5.113 src/web/statistics/datatier.c File Reference

```
#include "magma.h"
```

Defines

- `#define` [PORTAL_STATISTICS_TIMEOUT](#) 300

Functions

- `void` [statistics_init](#) (void)
Initialize the prepared sql statements used by the portal statistics page.
- `bool_t` [statistics_refresh](#) (void)
Refresh all portal statistics from the database, if they haven't been updated recently.

Variables

- `statistics_vp_t` [portal_stats](#) [12]
- `time_t` [statistics_last_updated](#) = 0
- `pthread_mutex_t` [portal_statistics_mutex](#) = PTHREAD_MUTEX_INITIALIZER

5.113.1 Define Documentation

5.113.1.1 `#define` [PORTAL_STATISTICS_TIMEOUT](#) 300

Definition at line 11 of file datatier.c.

Referenced by [statistics_refresh\(\)](#).

5.113.2 Function Documentation

5.113.2.1 `void` [statistics_init](#) (void)

Initialize the prepared sql statements used by the portal statistics page. datatier.c

Returns:

This function returns no value.

Definition at line 22 of file datatier.c.

References [mm_wipe\(\)](#), [portal_stat_emails_received_today](#), [portal_stat_emails_received_week](#), [portal_stat_emails_sent_today](#), [portal_stat_emails_sent_week](#), [portal_stat_total_users](#), [portal_stat_users_checked_email_today](#), [portal_stat_users_checked_email_week](#), [portal_stat_users_registered_today](#), [portal_stat_users_registered_total](#), [portal_stat_users_registered_week](#), [portal_stat_users_sent_email_today](#), [portal_stat_users_sent_email_week](#), and [statistics_vp_t::stmt](#).

Referenced by [statistics_refresh\(\)](#).

5.113.2.2 `bool_t statistics_refresh(void)`

Refresh all portal statistics from the database, if they haven't been updated recently.

Returns:

true if all statistics were refreshed successfully, or false if they were not.

Definition at line 44 of file `datatier.c`.

References `log_pedantic`, `mutex_lock()`, `mutex_unlock()`, `portal_statistics_mutex`, `PORTAL_STATISTICS_TIMEOUT`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `statistics_init()`, `statistics_last_updated`, `statistics_vp_t::stmt`, `stmt_get_result()`, and `statistics_vp_t::val`.

Referenced by `statistics_process()`.

5.113.3 Variable Documentation

5.113.3.1 `pthread_mutex_t portal_statistics_mutex = PTHREAD_MUTEX_INITIALIZER`

Definition at line 15 of file `datatier.c`.

Referenced by `statistics_refresh()`.

5.113.3.2 `statistics_vp_t portal_stats[12]`

Definition at line 13 of file `datatier.c`.

5.113.3.3 `time_t statistics_last_updated = 0`

Definition at line 14 of file `datatier.c`.

Referenced by `statistics_refresh()`.

5.114 src/web/teacher/datatier.c File Reference

```
#include "magma.h"
```

Functions

- void `teacher_data_free` (`teacher_data_t` *`teach`)
Free a spam signature object.
- void `teacher_data_delete` (`teacher_data_t` *`teach`)
Delete spam signature information from the database.
- `teacher_data_t` * `teacher_data_fetch` (uint64_t `signum`)
Fetch information about a spam signature from the database.
- void `teacher_data_save` (`teacher_data_t` *`teach`)
Save spam signature information to the cache.
- `teacher_data_t` * `teacher_data_get` (uint64_t `signum`)
Get information about a spam signature from the cache, or fall back to the database.

5.114.1 Function Documentation

5.114.1.1 void `teacher_data_delete` (`teacher_data_t` * `teach`)

Delete spam signature information from the database. `datatier.c`

HIGH: After training a signature, we should search the messages table for references to the signature being trained and update the message status flags. The `UPDATE_SIGNATURE_FLAGS_ADD/UPDATE_SIGNATURE_FLAGS_REMOVE` queries were created for that purpose but aren't being used right now. Note to self: retroactively brand messages as junk accordingly.

Note:

If the signature matched junk, all matching messages in the database belonging to the user will have their junk flag cleared. But if the signature didn't match junk, all matching messages in the database belonging to the user will have the junk flag added.

Parameters:

`teach` the spam signature to be removed from the database.

Returns:

This function returns no value.

Definition at line 38 of file `datatier.c`.

References `teacher_data_t::completed`, `teacher_data_t::disposition`, `log_pedantic`, `MAIL_MARK_JUNK`, `mm_wipe()`, `teacher_data_t::password`, `teacher_data_t::signature`, `teacher_data_t::signum`, `st_cleanup`, `stmt_exec()`, `teacher_data_t::username`, and `teacher_data_t::usernum`.

Referenced by `teacher_process()`.

5.114.1.2 **teacher_data_t* teacher_data_fetch (uint64_t *signum*)**

Fetch information about a spam signature from the database.

Parameters:

signum the numerical id of the spam signature to be retrieved.

NULL on failure, or a pointer to a newly allocated signature teacher object on success.

Definition at line 137 of file datatier.c.

References teacher_data_t::disposition, teacher_data_t::keynum, mm_alloc(), mm_wipe(), teacher_data_t::password, res_field_int8(), res_field_string(), res_field_uint64(), res_row_next(), res_table_free(), teacher_data_t::signature, teacher_data_t::signum, stmt_get_result(), teacher_data_free(), teacher_data_t::username, and teacher_data_t::usernum.

Referenced by teacher_data_get().

5.114.1.3 **void teacher_data_free (teacher_data_t * *teach*)**

Free a spam signature object.

Parameters:

teach the spam signature object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file datatier.c.

References mm_free(), teacher_data_t::password, teacher_data_t::signature, st_cleanup, and teacher_data_t::username.

Referenced by teacher_data_fetch(), teacher_data_get(), and teacher_process().

5.114.1.4 **teacher_data_t* teacher_data_get (uint64_t *signum*)**

Get information about a spam signature from the cache, or fall back to the database.

Parameters:

signum the numerical id of the spam signature to be retrieved.

NULL on failure, or a pointer to a newly allocated signature teacher object on success.

Definition at line 229 of file datatier.c.

References cache_get(), teacher_data_t::completed, data, deserialize_int32(), deserialize_st(), deserialize_uint64(), teacher_data_t::disposition, teacher_data_t::keynum, log_pedantic, mm_alloc(), mm_wipe(), teacher_data_t::password, PLACER, serialization_t, teacher_data_t::signature, teacher_data_t::signum, st_free(), teacher_data_fetch(), teacher_data_free(), teacher_data_t::username, and teacher_data_t::usernum.

Referenced by teacher_process().

5.114.1.5 **void teacher_data_save (teacher_data_t * *teach*)**

Save spam signature information to the cache.

Note:

The information will be cached for 2 hours.

Parameters:

teach the spam signature to be cached.

Returns:

This function returns no value.

Definition at line 189 of file datatier.c.

References `cache_set()`, `teacher_data_t::completed`, `data`, `teacher_data_t::disposition`, `teacher_data_t::keynum`, `log_pedantic`, `teacher_data_t::password`, `PLACER`, `serialize_int32()`, `serialize_st()`, `serialize_uint64()`, `teacher_data_t::signature`, `teacher_data_t::signum`, `st_free()`, `teacher_data_t::username`, and `teacher_data_t::usernum`.

Referenced by `teacher_process()`.

5.115 src/engine/config/global/global.c File Reference

```
#include "magma.h"
#include "keys.h"
```

Functions

- void [config_free](#) (void)
LOW: We should use use `basename()` and `dirname()` to cleanup path strings.
- void [config_output_value_generic](#) (chr_t *prefix, chr_t *name, M_TYPE type, void *val, bool_t required)
Output a key name and value in a generic way.
- void [config_output_value](#) (magma_keys_t *key)
Log the contents of a magma configuration option.
- void [config_output_settings](#) (void)
Log a display of all active magma configuration settings.
- void [config_output_help](#) (void)
Log all magma config key settings, as well as server, relay server, and cache server settings.
- bool_t [config_validate_settings](#) (void)
Validate all the user configuration settings.
- bool_t [config_value_set](#) (magma_keys_t *setting, stringer_t *value)
Set the value of a global config key.
- bool_t [config_load_defaults](#) (void)
Load all the default values for non-required configuration options.
- magma_keys_t * [config_key_lookup](#) (stringer_t *name)
Get a magma config key by name.
- bool_t [config_load_file_settings](#) (void)
Load the magma configuration file specified in `magma.config.file`, or from the command line.
- bool_t [config_load_database_settings](#) (void)
Load all magma configuration options present in the database.
- bool_t [config_load_cmdline_settings](#) (void)
Load all magma configuration options specified by the user on the command line.

Variables

- __thread char [threadBuffer](#) [1024]
- magma_t [magma](#) = { .config.file = "magma.config" }
- bool_t [exit_and_dump](#) = false
- stringer_t * [cmdline_config_data](#) = NULL

5.115.1 Function Documentation

5.115.1.1 void config_free (void)

LOW: We should use use `basename()` and `dirname()` to cleanup path strings. [global.c](#)

Free all loaded magma configuration options.

Note:

First all magma config keys will be freed, then the cache servers, relay servers, and magma servers.

Returns:

This function returns no value.

Definition at line 23 of file `global.c`.

References `magma_t::blacklists`, `cache_free()`, `log_pedantic`, `M_TYPE_BLOCK`, `M_TYPE_BOOLEAN`, `M_TYPE_DOUBLE`, `M_TYPE_ENUM`, `M_TYPE_FLOAT`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `magma_keys`, `mm_free()`, `magma_keys_t::norm`, `ns_free()`, `NULLER`, `PLACER`, `relay_free()`, `servers_free()`, `magma_t::smtp`, `st_cmp_cs_eq()`, `st_free()`, `store`, `type()`, and `multi_t::type`.

Referenced by `config_load_defaults()`, and `process_stop()`.

5.115.1.2 magma_keys_t* config_key_lookup (stringer_t * name)

Get a magma config key by name.

Parameters:

name a managed string with the name of the magma config option to be looked up.

Returns:

NULL on failure or a pointer to the found magma key object on success.

Definition at line 726 of file `global.c`.

References `magma_keys`, `NULLER`, and `st_cmp_ci_eq()`.

Referenced by `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_file_settings()`, and `config_validate_settings()`.

5.115.1.3 bool_t config_load_cmdline_settings (void)

Load all magma configuration options specified by the user on the command line.

Note:

Each key/value pair extracted from the database is submitted to the following logic: If a config option was loaded from the database, the key must allow it to be configurable via the database. Check to see that any key that has previously been set is allowed to be overwritten. If the key is required, it may not contain an empty value. Finally, this function sets the appropriate magma key corresponding to the config key. All leftover keys not matched to global magma keys will be configured via servers, relay, and cache server options.

Returns:

true if all database config options were parsed and evaluated successfully, or false on failure.

Definition at line 943 of file `global.c`.

References `cache_config()`, `cmdline_config_data`, `config_key_lookup()`, `config_value_set()`, `CONSTANT`, `magma_keys_t::file`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_key_next()`, `inx_cursor_t`, `inx_cursor_value_active()`, `log_critical`, `log_warn`, `mt_is_empty()`, `magma_keys_t::name`, `nvp_alloc()`, `nvp_free()`, `nvp_parse()`, `nvp_t::pairs`, `relay_config()`, `magma_keys_t::required`, `servers_config()`, `magma_keys_t::set`, `multi_t::st`, `st_char_get()`, `st_cmp_ci_starts()`, `st_empty`, `st_free()`, `st_length_int()`, and `multi_t::val`.

Referenced by `process_start()`.

5.115.1.4 `bool_t config_load_database_settings (void)`

Load all magma configuration options present in the database.

Note:

Each key/value pair extracted from the database is submitted to the following logic: If a config option was loaded from the database, the key must allow it to be configurable via the database. Check to see that any key that has previously been set is allowed to be overwritten. If the key is required, it may not contain an empty value. Finally, this function sets the appropriate magma key corresponding to the config key. All leftover keys not matched to global magma keys will be configured via servers, relay, and cache server options.

Returns:

true if all database config options were parsed and evaluated successfully, or false on failure.

Definition at line 850 of file `global.c`.

References `cache_config()`, `config_fetch_host_number()`, `config_fetch_settings()`, `config_key_lookup()`, `config_value_set()`, `CONSTANT`, `magma_keys_t::database`, `magma_t::host`, `log_critical`, `log_warn`, `magma_keys_t::name`, `magma_t::number`, `magma_keys_t::overwrite`, `PLACER`, `relay_config()`, `magma_keys_t::required`, `res_field_block()`, `res_field_length()`, `res_row_count()`, `res_row_get()`, `res_table_free()`, `servers_config()`, `magma_keys_t::set`, `st_char_get()`, `st_cmp_ci_eq()`, `st_cmp_ci_starts()`, `st_empty`, and `st_length_int()`.

Referenced by `process_start()`.

5.115.1.5 `bool_t config_load_defaults (void)`

Load all the default values for non-required configuration options.

Returns:

true if all default magma config values were successfully loaded, or false on failure.

Definition at line 701 of file `global.c`.

References `config_free()`, `config_value_set()`, `log_info`, and `magma_keys`.

Referenced by `args_parse()`, and `process_start()`.

5.115.1.6 `bool_t config_load_file_settings (void)`

Load the magma configuration file specified in `magma.config.file`, or from the command line.

Note:

Parses the config file data into a series of name/value pairs, and make sure that for each key: If a config option was loaded from the file, the key must allow it to be configurable via the file, and If the key is required, it may not contain an empty value. Finally, this function sets the appropriate magma key corresponding to the config key. All leftover keys not matched to global magma keys will be configured via servers, relay, and cache server options.

Returns:

true if all config file options were parsed and evaluated successfully, or false on failure.

Definition at line 748 of file global.c.

References `cache_config()`, `magma_t::config`, `config_key_lookup()`, `config_value_set()`, `CONSTANT`, `magma_keys_t::file`, `magma_t::file`, `file_load()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_key_next()`, `inx_cursor_t`, `inx_cursor_value_active()`, `log_critical`, `mt_is_empty()`, `magma_keys_t::name`, `nvp_alloc()`, `nvp_free()`, `nvp_parse()`, `nvp_t::pairs`, `relay_config()`, `magma_keys_t::required`, `servers_config()`, `magma_keys_t::set`, `multi_t::st`, `st_char_get()`, `st_cmp_ci_starts()`, `st_empty`, `st_free()`, `st_length_int()`, and `multi_t::val`.

Referenced by `process_start()`.

5.115.1.7 void config_output_help (void)

Log all magma config key settings, as well as server, relay server, and cache server settings.

Returns:

This function returns no value.

Definition at line 291 of file global.c.

References `multi_t::bl`, `cache_output_help()`, `config_output_value_generic()`, `log_info`, `log_options`, `M_LOG_FILE_DISABLE`, `M_LOG_FUNCTION_DISABLE`, `M_LOG_LINE_DISABLE`, `M_LOG_LINE_FEED_DISABLE`, `M_LOG_STACK_TRACE_DISABLE`, `M_LOG_TIME_DISABLE`, `magma_keys`, `magma_keys_t::norm`, `relay_output_help()`, `magma_keys_t::required`, `servers_output_help()`, and `multi_t::val`.

Referenced by `args_parse()`.

5.115.1.8 void config_output_settings (void)

Log a display of all active magma configuration settings.

Returns:

This function returns no value.

Definition at line 264 of file global.c.

References `cache_output_settings()`, `magma_t::config`, `config_output_value()`, `magma_t::file`, `magma_t::host`, `log_info`, `magma_keys`, `magma_t::name`, `magma_t::number`, `relay_output_settings()`, and `servers_output_settings()`.

Referenced by `config_validate_settings()`.

5.115.1.9 void config_output_value (magma_keys_t * key)

Log the contents of a magma configuration option.

Parameters:

key a pointer to the magma configuration key to be dumped.

Returns:

This function returns no value.

Definition at line 192 of file global.c.

References `magma_t::blacklists`, `log_info`, `log_pedantic`, `M_TYPE_BOOLEAN`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `magma_keys_t::name`, `magma_keys_t::norm`, `ns_empty()`, `NULLER`, `PLACER`, `magma_t::smtp`, `st_char_get()`, `st_cmp_cs_eq()`, `st_empty`, `st_length_int()`, `magma_keys_t::store`, and `multi_t::type`.

Referenced by `config_output_settings()`.

5.115.1.10 void config_output_value_generic (chr_t * *prefix*, chr_t * *name*, M_TYPE *type*, void * *val*, bool_t *required*)

Output a key name and value in a generic way.

Parameters:

- prefix* a pointer to a null-terminated string containing an optional prefix to be printed before the supplied key name.
- name* a pointer to a null-terminated string containing the name of the key being output.
- type* an M_TYPE value specifying the multi-type of the key value.
- val* a void pointer to the value of the specified key, which will be printed in accordance with the supplied multi-type.
- required* a boolean value specifying whether the specified key is a required configuration option.

Returns:

This function returns no value.

Definition at line 78 of file global.c.

References magma_t::blacklists, CONSTANT, DMTP, GENERIC, HTTP, IMAP, log_info, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_ENUM, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, MOLTEN, ns_empty(), NULLER, PLACER, POP, SMTP, magma_t::smtp, st_char_get(), st_cmp_cs_eq(), st_empty, st_length_int(), SUBMISSION, TCP_PORT, and TLS_PORT.

Referenced by cache_output_help(), config_output_help(), relay_output_help(), and servers_output_help().

5.115.1.11 bool_t config_validate_settings (void)

Validate all the user configuration settings.

Note:

The steps are as follows: 1. Check to see that all required keys have been set. 2. If magma.iface.virus.available is set, magma.iface.virus.signatures must be set. 3. Make sure $10 \leq \text{magma.iface.cache.retry} \leq 86400$ 4. Make sure $1 \leq \text{magma.iface.cache.timeout} \leq 3600$ 5. Make sure $\text{magma.iface.cache.retry} \leq \text{magma.iface.cache.timeout}$ 6. Make sure $40 \leq \text{magma.smtp.wrap_line_length} \leq 65535$ 7. Make sure $8 \leq \text{magma.smtp.recipient_limit} \leq 32768$ 8. Make sure $16 \leq \text{magma.smtp.relay_limit}$ 9. Make sure $16384 \leq \text{system_ulimit_max}(\text{RLIMIT_STACK})$ 10. If magma.system.daemonize is set, make sure magma.output.file is not false 11. If magma.output.file is enabled, magma.output.path must be set. 12. If [magma.dkim.enabled](#) is set, then magma.dkim.domain, magma.dkim.selector, and magma.dkim.key must all be set. 13. Validate all the configured magma servers, relay servers, and cache servers. 14. Check all config key filenames and directories to ensure that they exist and are accessible. 15. Make sure magma.admin.contact and point to valid email addresses, if they are specified. 16. If magma.config.output_config is set, dump the current configuration.

Definition at line 329 of file global.c.

References magma_t::abuse, magma_t::admin, magma_t::cache, cache_validate(), magma_t::config, CONFIG_CHECK_DIR_READABLE, CONFIG_CHECK_DIR_READWRITE, CONFIG_CHECK_FILE_READABLE, config_key_lookup(), config_output_settings(), magma_t::contact, contact_business_valid_email(), magma_t::daemonize, magma_t::dime, magma_t::dkim, magma_t::domain, magma_t::enabled, exit_and_dump, magma_t::file, magma_t::http, magma_t::iface, magma_t::key, magma_t::library, log_critical, log_warn, magma_keys, magma_t::minimum_password_length, magma_t::output, magma_t::output_config, magma_t::path, PLACER, magma_t::recipient_limit, magma_t::relay_limit, relay_validate(), magma_t::root_directory, magma_t::secure, magma_t::selector, servers_validate(), magma_keys_t::set, magma_t::signet, magma_t::smtp, magma_t::spool, st_char_get(), magma_t::system, system_ulimit_max(), magma_t::thread_stack_size, magma_t::virus, and magma_t::wrap_line_length.

Referenced by process_start().

5.115.1.12 bool_t config_value_set (magma_keys_t * *setting*, stringer_t * *value*)

Set the value of a global config key.

Note:

This function will also free the value of the global config key if it has already previously been set.

Parameters:

setting a pointer to the global key to have its value adjusted.

value a managed string containing the new key value, or if NULL, the key's default value will be used.

Returns:

true if the specified key's value was set successfully, or false on failure.

LOW: Realtime blacklist domains are handled using custom code because we don't yet have a generic type to store lists.

Definition at line 515 of file global.c.

References multi_t::binary, magma_t::blacklists, CONSTANT, CONTIGUOUS, HEAP, multi_t::i32, multi_t::i64, multi_t::i8, int32_conv_st(), int64_conv_st(), int8_conv_st(), log_critical, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, MAGMA_BLACKLIST_INSTANCES, MANAGED_T, magma_keys_t::name, magma_keys_t::norm, multi_t::ns, ns_dupe(), ns_empty(), ns_free(), ns_import(), NULLER, PLACER, magma_t::smtp, smtp_add_bypass_entry(), multi_t::st, st_char_get(), st_cmp_ci_eq(), st_cmp_cs_eq(), st_dupe_opts(), st_empty, st_free(), st_length_get(), magma_keys_t::store, multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, uint16_conv_st(), uint32_conv_st(), uint64_conv_st(), uint8_conv_st(), and multi_t::val.

Referenced by config_load_cmdline_settings(), config_load_database_settings(), config_load_defaults(), and config_load_file_settings().

5.115.2 Variable Documentation

5.115.2.1 stringer_t* cmdline_config_data = NULL

Definition at line 14 of file global.c.

Referenced by args_parse(), and config_load_cmdline_settings().

5.115.2.2 bool_t exit_and_dump = false

Definition at line 13 of file global.c.

Referenced by args_parse(), config_validate_settings(), and process_start().

5.115.2.3 magma_t magma = { .config.file = "magma.config" }

Definition at line 12 of file global.c.

Referenced by api_response(), args_parse(), auth_legacy(), auth_login(), auth_sanitise_username(), cache_alloc(), cache_free(), cache_output_settings(), cache_start(), cache_stop(), cache_validate(), color_supported(), con_init_network_buffer(), config_fetch_host_number(), config_fetch_settings(), contact_business(), dh_params_generate(), dkim_signature_create(), dkim_start(), http_body(), http_content_load_fonts(), http_content_refresh(), http_content_start(), http_data_value_parse(), http_load_file(), http_parse_header(), http_parse_method(), http_print_301(), http_response(), http_response_connection(), http_response_header(), http_response_options(), imap_append_message(), imap_command_parser(), jansson_flags(), json_api_dispatch(), lib_load(), lib_unload(), log_internal(), log_rotate(), log_start(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_create_directory(), mail_db_insert_duplicate_message(), mail_db_insert_message(), mail_message_cleanup(), mail_message_path(), mm_sec_start(), net_init(), net_listen(), net_trigger(), portal_endpoint(), portal_endpoint_error(), portal_endpoint_response(), portal_process(), portal_upload(), prime_start(), process_start(), process_stop(), queue_init(), queue_shutdown(), queue_signal(), rand_start(), rand_thread_start(), register_business_step1(), register_business_step2(), register_business_validate_password(), register_captcha_random_font(), register_data_insert_user(), relay_alloc(), relay_counter(), relay_free(), relay_output_settings(), relay_validate(), servers_alloc(), servers_encryption_start(), servers_encryption_stop(), servers_free(), servers_get_by_protocol(), servers_get_by_socket(), servers_get_count_using_port(), servers_network_start(), servers_network_stop(), servers_output_settings(), servers_validate(), sess_create(), sess_get(), sess_token(), smtp_accept_message(), smtp_add_bypass_entry(), smtp_

bounce(), smtp_bypass_check(), smtp_check_rbl(), smtp_client_connect(), smtp_client_send_helo(), smtp_data(), smtp_ehlo(), smtp_mail_from(), smtp_parse_helo_domain(), smtp_parse_mail_from_path(), smtp_parse_rcpt_to(), smtp_rcpt_to(), spf_start(), spf_stop(), spool_path(), sql_open(), sql_start(), sql_stop(), st_alloc_opts(), st_realloc(), stmt_start(), stmt_stop(), system_change_root_directory(), system_fork_daemon(), system_init_core_dumps(), system_init_impersonation(), system_init_resource_limits(), tank_start(), thread_launch(), tls_server_create(), virus_check(), virus_engine_create(), virus_engine_refresh(), virus_sigs_total(), virus_start(), and virus_stop().

5.115.2.4 __thread char threadBuffer[1024]

Definition at line 11 of file global.c.

5.116 src/engine/config/global/global.h File Reference

Data Structures

- struct [magma_keys_t](#)
- struct [magma_t](#)

Functions

- [uint64_t config_fetch_host_number](#) (void)
datatier.c
- [table_t * config_fetch_settings](#) (void)
Retrieve the entire collection of configuration key/value pairs from the database.
- [void config_free](#) (void)
global.c
- [magma_keys_t * config_key_lookup](#) ([stringer_t](#) *name)
Get a magma config key by name.
- [bool_t config_load_database_settings](#) (void)
Load all magma configuration options present in the database.
- [bool_t config_load_cmdline_settings](#) (void)
Load all magma configuration options specified by the user on the command line.
- [bool_t config_load_defaults](#) (void)
Load all the default values for non-required configuration options.
- [bool_t config_load_file_settings](#) (void)
Load the magma configuration file specified in magma.config.file, or from the command line.
- [void config_output_help](#) (void)
Log all magma config key settings, as well as server, relay server, and cache server settings.
- [void config_output_settings](#) (void)
Log a display of all active magma configuration settings.
- [void config_output_value_generic](#) ([chr_t](#) *prefix, [chr_t](#) *name, [M_TYPE](#) type, void *val, [bool_t](#) required)
Output a key name and value in a generic way.
- [void config_output_value](#) ([magma_keys_t](#) *key)
Log the contents of a magma configuration option.
- [bool_t config_validate_settings](#) (void)
Validate all the user configuration settings.
- [bool_t config_value_set](#) ([magma_keys_t](#) *setting, [stringer_t](#) *value)
Set the value of a global config key.

5.116.1 Function Documentation

5.116.1.1 uint64_t config_fetch_host_number (void)

datatier.c datatier.c

Returns:

this host's numerical identifier, or 0 on failure.

Definition at line 14 of file datatier.c.

References magma_t::host, log_error, magma, mm_wipe(), magma_t::name, ns_length_get(), res_field_uint64(), res_row_next(), res_table_free(), and stmt_get_result().

Referenced by config_load_database_settings().

5.116.1.2 table_t* config_fetch_settings (void)

Retrieve the entire collection of configuration key/value pairs from the database.

Parameters:

none This function accepts no parameters.

Returns:

NULL on failure, or a database table of configuration key/value pairs on success.

Definition at line 40 of file datatier.c.

References magma_t::host, magma, mm_wipe(), magma_t::name, ns_length_get(), and stmt_get_result().

Referenced by config_load_database_settings().

5.116.1.3 void config_free (void)

[global.c](#) [global.c](#)

Free all loaded magma configuration options.

Note:

First all magma config keys will be freed, then the cache servers, relay servers, and magma servers.

Returns:

This function returns no value.

Definition at line 23 of file global.c.

References magma_t::blacklists, cache_free(), log_pedantic, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_ENUM, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma_keys, mm_free(), magma_keys_t::norm, ns_free(), NULLER, PLACER, relay_free(), servers_free(), magma_t::smtp, st_cmp_cs_eq(), st_free(), store, type(), and multi_t::type.

Referenced by config_load_defaults(), and process_stop().

5.116.1.4 magma_keys_t* config_key_lookup (stringer_t * name)

Get a magma config key by name.

Parameters:

name a managed string with the name of the magma config option to be looked up.

Returns:

NULL on failure or a pointer to the found magma key object on success.

Definition at line 726 of file global.c.

References magma_keys, NULLER, and st_cmp_ci_eq().

Referenced by config_load_cmdline_settings(), config_load_database_settings(), config_load_file_settings(), and config_validate_settings().

5.116.1.5 bool_t config_load_cmdline_settings (void)

Load all magma configuration options specified by the user on the command line.

Note:

Each key/value pair extracted from the database is submitted to the following logic: If a config option was loaded from the database, the key must allow it to be configurable via the database. Check to see that any key that has previously been set is allowed to be overwritten. If the key is required, it may not contain an empty value. Finally, this function sets the appropriate magma key corresponding to the config key. All leftover keys not matched to global magma keys will be configured via servers, relay, and cache server options.

Returns:

true if all database config options were parsed and evaluated successfully, or false on failure.

Definition at line 943 of file global.c.

References cache_config(), cmdline_config_data, config_key_lookup(), config_value_set(), CONSTANT, magma_keys_t::file, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_key_next(), inx_cursor_t, inx_cursor_value_active(), log_critical, log_warn, mt_is_empty(), magma_keys_t::name, nvp_alloc(), nvp_free(), nvp_parse(), nvp_t::pairs, relay_config(), magma_keys_t::required, servers_config(), magma_keys_t::set, multi_t::st, st_char_get(), st_cmp_ci_starts(), st_empty, st_free(), st_length_int(), and multi_t::val.

Referenced by process_start().

5.116.1.6 bool_t config_load_database_settings (void)

Load all magma configuration options present in the database.

Note:

Each key/value pair extracted from the database is submitted to the following logic: If a config option was loaded from the database, the key must allow it to be configurable via the database. Check to see that any key that has previously been set is allowed to be overwritten. If the key is required, it may not contain an empty value. Finally, this function sets the appropriate magma key corresponding to the config key. All leftover keys not matched to global magma keys will be configured via servers, relay, and cache server options.

Returns:

true if all database config options were parsed and evaluated successfully, or false on failure.

Definition at line 850 of file global.c.

References cache_config(), config_fetch_host_number(), config_fetch_settings(), config_key_lookup(), config_value_set(), CONSTANT, magma_keys_t::database, magma_t::host, log_critical, log_warn, magma_keys_t::name, magma_t::number, magma_keys_t::overwrite,

PLACER, relay_config(), magma_keys_t::required, res_field_block(), res_field_length(), res_row_count(), res_row_get(), res_table_free(), servers_config(), magma_keys_t::set, st_char_get(), st_cmp_ci_eq(), st_cmp_ci_starts(), st_empty, and st_length_int().

Referenced by process_start().

5.116.1.7 bool_t config_load_defaults (void)

Load all the default values for non-required configuration options.

Returns:

true if all default magma config values were successfully loaded, or false on failure.

Definition at line 701 of file global.c.

References config_free(), config_value_set(), log_info, and magma_keys.

Referenced by args_parse(), and process_start().

5.116.1.8 bool_t config_load_file_settings (void)

Load the magma configuration file specified in magma.config.file, or from the command line.

Note:

Parses the config file data into a series of name/value pairs, and make sure that for each key: If a config option was loaded from the file, the key must allow it to be configurable via the file, and If the key is required, it may not contain an empty value. Finally, this function sets the appropriate magma key corresponding to the config key. All leftover keys not matched to global magma keys will be configured via servers, relay, and cache server options.

Returns:

true if all config file options were parsed and evaluated successfully, or false on failure.

Definition at line 748 of file global.c.

References cache_config(), magma_t::config, config_key_lookup(), config_value_set(), CONSTANT, magma_keys_t::file, magma_t::file, file_load(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_key_next(), inx_cursor_t, inx_cursor_value_active(), log_critical, mt_is_empty(), magma_keys_t::name, nvp_alloc(), nvp_free(), nvp_parse(), nvp_t::pairs, relay_config(), magma_keys_t::required, servers_config(), magma_keys_t::set, multi_t::st, st_char_get(), st_cmp_ci_starts(), st_empty, st_free(), st_length_int(), and multi_t::val.

Referenced by process_start().

5.116.1.9 void config_output_help (void)

Log all magma config key settings, as well as server, relay server, and cache server settings.

Returns:

This function returns no value.

Definition at line 291 of file global.c.

References multi_t::bl, cache_output_help(), config_output_value_generic(), log_info, log_options, M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE_DISABLE, M_LOG_TIME_DISABLE, magma_keys, magma_keys_t::norm, relay_output_help(), magma_keys_t::required, servers_output_help(), and multi_t::val.

Referenced by args_parse().

5.116.1.10 void config_output_settings (void)

Log a display of all active magma configuration settings.

Returns:

This function returns no value.

Definition at line 264 of file global.c.

References cache_output_settings(), magma_t::config, config_output_value(), magma_t::file, magma_t::host, log_info, magma_keys, magma_t::name, magma_t::number, relay_output_settings(), and servers_output_settings().

Referenced by config_validate_settings().

5.116.1.11 void config_output_value (magma_keys_t * key)

Log the contents of a magma configuration option.

Parameters:

key a pointer to the magma configuration key to be dumped.

Returns:

This function returns no value.

Definition at line 192 of file global.c.

References magma_t::blacklists, log_info, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma_keys_t::name, magma_keys_t::norm, ns_empty(), NULLER, PLACER, magma_t::smtp, st_char_get(), st_cmp_cs_eq(), st_empty, st_length_int(), magma_keys_t::store, and multi_t::type.

Referenced by config_output_settings().

5.116.1.12 void config_output_value_generic (chr_t * prefix, chr_t * name, M_TYPE type, void * val, bool_t required)

Output a key name and value in a generic way.

Parameters:

prefix a pointer to a null-terminated string containing an optional prefix to be printed before the supplied key name.

name a pointer to a null-terminated string containing the name of the key being output.

type an M_TYPE value specifying the multi-type of the key value.

val a void pointer to the value of the specified key, which will be printed in accordance with the supplied multi-type.

required a boolean value specifying whether the specified key is a required configuration option.

Returns:

This function returns no value.

Definition at line 78 of file global.c.

References magma_t::blacklists, CONSTANT, DMTP, GENERIC, HTTP, IMAP, log_info, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_ENUM, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, MOLTEN, ns_empty(), NULLER, PLACER, POP, SMTP, magma_t::smtp, st_char_get(), st_cmp_cs_eq(), st_empty, st_length_int(), SUBMISSION, TCP_PORT, and TLS_PORT.

Referenced by cache_output_help(), config_output_help(), relay_output_help(), and servers_output_help().

5.116.1.13 `bool_t config_validate_settings(void)`

Validate all the user configuration settings.

Note:

The steps are as follows: 1. Check to see that all required keys have been set. 2. If `magma.iface.virus.available` is set, `magma.iface.virus.signatures` must be set. 3. Make sure `10 <= magma.iface.cache.retry <= 86400` 4. Make sure `1 <= magma.iface.cache.timeout <= 3600` 5. Make sure `magma.iface.cache.retry <= magma.iface.cache.timeout` 6. Make sure `40 <= magma.smtp.wrap_line_length <= 65535` 7. Make sure `8 <= magma.smtp.recipient_limit <= 32768` 8. Make sure `16 <= magma.smtp.relay_limit` 9. Make sure `16384 <= system_ulimit_max(RLIMIT_STACK)` 10. If `magma.system.daemonize` is set, make sure `magma.output.file` is not false 11. If `magma.output.file` is enabled, `magma.output.path` must be set. 12. If `magma.dkim.enabled` is set, then `magma.dkim.domain`, `magma.dkim.selector`, and `magma.dkim.key` must all be set. 13. Validate all the configured magma servers, relay servers, and cache servers. 14. Check all config key filenames and directories to ensure that they exist and are accessible. 15. Make sure `magma.admin.contact` and point to valid email addresses, if they are specified. 16. If `magma.config.output_config` is set, dump the current configuration.

Definition at line 329 of file `global.c`.

References `magma_t::abuse`, `magma_t::admin`, `magma_t::cache`, `cache_validate()`, `magma_t::config`, `CONFIG_CHECK_DIR_READABLE`, `CONFIG_CHECK_DIR_READWRITE`, `CONFIG_CHECK_FILE_READABLE`, `config_key_lookup()`, `config_output_settings()`, `magma_t::contact`, `contact_business_valid_email()`, `magma_t::daemonize`, `magma_t::dime`, `magma_t::dkim`, `magma_t::domain`, `magma_t::enabled`, `exit_and_dump`, `magma_t::file`, `magma_t::http`, `magma_t::iface`, `magma_t::key`, `magma_t::library`, `log_critical`, `log_warn`, `magma_keys`, `magma_t::minimum_password_length`, `magma_t::output`, `magma_t::output_config`, `magma_t::path`, `PLACER`, `magma_t::recipient_limit`, `magma_t::relay_limit`, `relay_validate()`, `magma_t::root_directory`, `magma_t::secure`, `magma_t::selector`, `servers_validate()`, `magma_keys_t::set`, `magma_t::signet`, `magma_t::smtp`, `magma_t::spool`, `st_char_get()`, `magma_t::system`, `system_ulimit_max()`, `magma_t::thread_stack_size`, `magma_t::virus`, and `magma_t::wrap_line_length`.

Referenced by `process_start()`.

5.116.1.14 `bool_t config_value_set(magma_keys_t *setting, stringer_t *value)`

Set the value of a global config key.

Note:

This function will also free the value of the global config key if it has already previously been set.

Parameters:

setting a pointer to the global key to have its value adjusted.

value a managed string containing the new key value, or if NULL, the key's default value will be used.

Returns:

true if the specified key's value was set successfully, or false on failure.

LOW: Realtime blacklist domains are handled using custom code because we don't yet have a generic type to store lists.

Definition at line 515 of file `global.c`.

References `multi_t::binary`, `magma_t::blacklists`, `CONSTANT`, `CONTIGUOUS`, `HEAP`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `int32_conv_st()`, `int64_conv_st()`, `int8_conv_st()`, `log_critical`, `M_TYPE_BOOLEAN`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `MAGMA_BLACKLIST_INSTANCES`, `MANAGED_T`, `magma_keys_t::name`, `magma_keys_t::norm`, `multi_t::ns`, `ns_dupe()`, `ns_empty()`, `ns_free()`, `ns_import()`, `NULLER`, `PLACER`, `magma_t::smtp`, `smtp_add_bypass_entry()`, `multi_t::st`, `st_char_get()`, `st_cmp_ci_eq()`, `st_cmp_cs_eq()`, `st_dupe_opts()`, `st_empty`, `st_free()`, `st_length_get()`, `magma_keys_t::store`, `multi_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, `uint16_conv_st()`, `uint32_conv_st()`, `uint64_conv_st()`, `uint8_conv_st()`, and `multi_t::val`.

Referenced by `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_defaults()`, and `config_load_file_settings()`.

5.117 src/engine/config/relay/relay.c File Reference

```
#include "magma.h"
#include "keys.h"
```

Functions

- void `relay_counter` (void)
Update the relay server counters to reflect the number of configured standard and premium relay servers.
- void `relay_free` (void)
Free magma's relay server configuration options.
- `relay_t * relay_alloc` (uint32_t *number*)
Allocate and initialize a magma server relay object, or return it if it already exists.
- `bool_t relay_validate` (void)
Validate all of the configured magma relay server instances.
- void `relay_output_settings` (void)
- `bool_t relay_set_value` (relay_keys_t **setting*, relay_t **relay*, stringer_t **value*)
Set a specified key for a relay server configuration entry.
- `bool_t relay_config` (stringer_t **name*, stringer_t **value*)
Set the value of a relay server configuration entry by name.
- void `relay_output_help` (void)
Log all relay key information to be returned via "magma -h".

5.117.1 Function Documentation

5.117.1.1 relay_t* relay_alloc (uint32_t *number*)

Allocate and initialize a magma server relay object, or return it if it already exists.

Parameters:

number a zero-based index into the global relay server table where the entry should be created and/or queried.

Returns:

NULL on failure, or a pointer to the requested relay server object on success.

Definition at line 86 of file relay.c.

References magma_t::host, log_critical, magma, mm_alloc(), magma_t::relay, relay_keys, and relay_set_value().

Referenced by relay_config().

5.117.1.2 bool_t relay_config (stringer_t * *name*, stringer_t * *value*)

Set the value of a relay server configuration entry by name.

Note:

This function is designed to operate on lines from a configuration source, like a file or database. When a named server is referenced for the first time, a new relay server configuration instance will be allocated.

Parameters:

name a managed string containing the human-readable relay server configuration parameter to be set.

value a managed string containing the new value of the relay server config key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 448 of file relay.c.

References `bracket_extract_pl()`, `CONSTANT`, `log_critical`, `MAGMA_RELAY_INSTANCES`, `NULLER`, `pl_char_get()`, `pl_empty()`, `pl_length_get()`, `PLACER`, `placer_t`, `relay_alloc()`, `relay_keys`, `relay_set_value()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_cmp_ci_starts()`, `st_length_get()`, `st_length_int()`, and `uint32_conv_bl()`.

Referenced by `config_load_cmdline_settings()`, `config_load_database_settings()`, and `config_load_file_settings()`.

5.117.1.3 void relay_counter (void)

Update the relay server counters to reflect the number of configured standard and premium relay servers.

Returns:

This function returns no value.

Definition at line 16 of file relay.c.

References `magma_t::count`, `magma_t::host`, `magma`, `MAGMA_RELAY_INSTANCES`, `magma_t::premium`, and `magma_t::relay`.

Referenced by `relay_set_value()`.

5.117.1.4 void relay_free (void)

Free magma's relay server configuration options.

Note:

This function assumes all worker threads have finished, and should only be called during the normal shutdown process.

Parameters:

This function returns no value.

Definition at line 39 of file relay.c.

References `magma_t::host`, `log_pedantic`, `M_TYPE_BLOCK`, `M_TYPE_BOOLEAN`, `M_TYPE_DOUBLE`, `M_TYPE_ENUM`, `M_TYPE_FLOAT`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `magma`, `MAGMA_RELAY_INSTANCES`, `mm_free()`, `relay_keys_t::norm`, `ns_empty()`, `ns_free()`, `relay_keys_t::offset`, `magma_t::relay`, `relay_keys`, `st_empty`, `st_free()`, `type()`, and `multi_t::type`.

Referenced by `config_free()`.

5.117.1.5 void relay_output_help (void)

Log all relay key information to be returned via "magma -h".

Returns:

This function returns no value.

Definition at line 508 of file relay.c.

References multi_t::bl, config_output_value_generic(), log_info, log_options, M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE_DISABLE, M_LOG_TIME_DISABLE, relay_keys_t::norm, relay_keys, relay_keys_t::required, and multi_t::val.

Referenced by config_output_help().

5.117.1.6 void relay_output_settings (void)

Definition at line 228 of file relay.c.

References magma_t::host, log_info, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_RELAY_INSTANCES, relay_keys_t::norm, ns_empty(), relay_keys_t::offset, magma_t::relay, relay_keys, st_char_get(), st_empty, st_length_int(), multi_t::type, and type().

Referenced by config_output_settings().

5.117.1.7 bool_t relay_set_value (relay_keys_t * setting, relay_t * relay, stringer_t * value)

Set a specified key for a relay server configuration entry.

Note:

This function will allocate space for a copy of the key value, and return an error if it is not able to convert it to the proper key data type.

Parameters:

setting a pointer to the relay server configuration key to be set.

relay a pointer to the relay server configuration entry to be adjusted.

value a managed string containing the new value of the config key.

Returns:

true if the value was successfully set, or false on error.

Definition at line 303 of file relay.c.

References multi_t::binary, CONSTANT, CONTIGUOUS, HEAP, multi_t::i32, multi_t::i64, multi_t::i8, int32_conv_st(), int64_conv_st(), int8_conv_st(), log_critical, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, MANAGED_T, relay_keys_t::name, relay_keys_t::norm, multi_t::ns, ns_dup(), ns_empty(), ns_free(), ns_import(), relay_keys_t::offset, relay_counter(), multi_t::st, st_char_get(), st_cmp_ci_eq(), st_dup_opts(), st_empty, st_free(), st_length_get(), type(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, uint16_conv_st(), uint32_conv_st(), uint64_conv_st(), uint8_conv_st(), and multi_t::val.

Referenced by relay_alloc(), and relay_config().

5.117.1.8 bool_t relay_validate (void)

Validate all of the configured magma relay server instances. Iterates through all of the server structures and verifies that each was supplied with valid values for all of its required keys. This function also applies the application specific validation rules. Specifically it verifies that only one server structure has been configured to use a given port number.

Note:

This makes set that all required config keys have been set, and that at least one host has been configured.

Returns:

true if all relay servers have been validated, or false on failure.

Definition at line 115 of file relay.c.

References magma_t::host, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, log_critical, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_RELAY_INSTANCES, relay_keys_t::norm, ns_empty(), relay_keys_t::offset, magma_t::relay, relay_keys, st_empty, multi_t::type, type(), multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by config_validate_settings().

5.118 src/servers/smtp/relay.c File Reference

```
#include "magma.h"
```

Functions

- void `smtp_client_close` (`client_t` *client)
Issue an smtp client QUIT command.
- `client_t` * `smtp_client_connect` (`int_t` premium)
Connect to a randomly selected mail relay server, and wait for a successful banner message.
- `int_t` `smtp_client_send_helo` (`client_t` *client)
Issue a EHLO command to an smtp server, or fall back to HELO, and wait for a successful response.
- `int_t` `smtp_client_send_mailfrom` (`client_t` *client, `stringer_t` *mailfrom, `size_t` send_size)
Issue a MAIL FROM command to an smtp server, and wait for a successful response.
- `int_t` `smtp_client_send_nullfrom` (`client_t` *client)
Issue a MAIL FROM command to an smtp server with a null sender, and wait for a successful response.
- `int_t` `smtp_client_send_rcptto` (`client_t` *client, `stringer_t` *rcptto)
Issue a RCPT TO command to an smtp server, and wait for a successful response.
- `int_t` `smtp_client_send_data` (`client_t` *client, `stringer_t` *message, `bool_t` dotstuffed)
Issue a DATA command to an smtp server, and wait for a successful response.

5.118.1 Function Documentation

5.118.1.1 void smtp_client_close (client_t * client)

Issue an smtp client QUIT command. relay.c

Parameters:

client a pointer to the smtp client session to be closed.

Returns:

This function returns no value.

Definition at line 15 of file relay.c.

References `client_close()`, `client_read_line()`, `client_write()`, and `PLACER`.

Referenced by `portal_smtp_relay_message()`, `smtp_bounce()`, `smtp_forward_message()`, `smtp_relay_message()`, `smtp_reply()`, and `smtp_send_message()`.

5.118.1.2 client_t* smtp_client_connect (int_t premium)

Connect to a randomly selected mail relay server, and wait for a successful banner message.

Parameters:

premium if set, a premium relay will be selected instead of a standard one.

Returns:

NULL on failure or a pointer to the newly established network client object connected to a mail relay on success.

Definition at line 31 of file relay.c.

References `client_close()`, `client_connect()`, `client_read_line()`, `client_secure()`, `client_t`, `magma_t::count`, `magma_t::host`, `log_pedantic`, `magma`, `MAGMA_RELAY_INSTANCES`, `relay_t::name`, `net_set_timeout()`, `relay_t::port`, `magma_t::premium`, `rand_get_uint32()`, `magma_t::relay`, `relay_t::secure`, `st_char_get()`, `st_length_get()`, `st_length_int()`, and `magma_t::timeout`.

Referenced by `portal_smtp_relay_message()`, `smtp_bounce()`, `smtp_forward_message()`, `smtp_relay_message()`, `smtp_reply()`, and `smtp_send_message()`.

5.118.1.3 int_t smtp_client_send_data (client_t * client, stringer_t * message, bool_t dotstuffed)

Issue a DATA command to an smtp server, and wait for a successful response.

Parameters:

client a pointer to the network client to issue the DATA command.

message a pointer to a managed string containing the body of the message to be sent.

dotstuffed a boolean to where true indicates the supplied message has already been dotstuffed.

Returns:

-3 for internal errors, -2 if the remote server rejected the command, -1 on general network failure, or 1 on success.

Definition at line 243 of file relay.c.

References `client_read_line()`, `client_write()`, `HEAP`, `JOINTED`, `log_pedantic`, `MANAGEDBUF`, `MAPPED_T`, `pl_char_get()`, `pl_length_int()`, `pl_starts_with_char()`, `PLACER`, `st_char_get()`, `st_cleanup`, `st_cmp_cs_ends()`, `st_dupe_opts()`, `st_empty`, `st_length_get()`, `st_length_int()`, `st_quick()`, and `st_replace()`.

Referenced by `portal_smtp_relay_message()`, `smtp_bounce()`, `smtp_forward_message()`, `smtp_relay_message()`, `smtp_reply()`, and `smtp_send_message()`.

5.118.1.4 int_t smtp_client_send_helo (client_t * client)

Issue a EHLO command to an smtp server, or fall back to HELO, and wait for a successful response.

Parameters:

client a pointer to the network client to issue the remote command.

Returns:

-1 on failure or 1 on success.

Definition at line 110 of file relay.c.

References `client_print()`, `client_read_line()`, `magma_t::domain`, `magma_t::host`, `log_pedantic`, `magma`, `magma_t::name`, `ns_length_get()`, `pl_init()`, `pl_null()`, `placer_t`, `st_char_get()`, `st_empty`, `st_length_get()`, and `magma_t::system`.

Referenced by `portal_smtp_relay_message()`, `smtp_bounce()`, `smtp_forward_message()`, `smtp_relay_message()`, `smtp_reply()`, and `smtp_send_message()`.

5.118.1.5 int_t smtp_client_send_mailfrom (client_t * client, stringer_t * mailfrom, size_t send_size)

Issue a MAIL FROM command to an smtp server, and wait for a successful response.

Parameters:

client a pointer to the network client to issue the MAIL FROM command.
mailfrom a pointer to a managed string containing the address parameter for the MAIL FROM command.
send_size if greater than 0, specify the optional SIZE parameter to the MAIL FROM command.

Returns:

-2 if the remote server rejected the command, -1 on general network failure, or 1 on success.

LOW: Technically we should only be sending the size parameter if the EHLO response indicates support.

Definition at line 168 of file relay.c.

References client_print(), client_read_line(), log_pedantic, st_char_get(), st_length_get(), and st_length_int().

Referenced by portal_smtp_relay_message(), smtp_forward_message(), smtp_relay_message(), and smtp_send_message().

5.118.1.6 int_t smtp_client_send_nullfrom (client_t * client)

Issue a MAIL FROM command to an smtp server with a null sender, and wait for a successful response.

Note:

Null senders are used when the sender is not concerned about being notified about bounced messages.

Parameters:

client a pointer to the network client to issue the MAIL FROM command.

Returns:

-2 if the remote server rejected the command, -1 on general network failure, or 1 on success.

Definition at line 196 of file relay.c.

References client_print(), client_read_line(), log_pedantic, st_char_get(), and st_length_int().

Referenced by smtp_bounce(), and smtp_reply().

5.118.1.7 int_t smtp_client_send_rcptto (client_t * client, stringer_t * rcptto)

Issue a RCPT TO command to an smtp server, and wait for a successful response.

Parameters:

client a pointer to the network client to issue the RCPT TO command.
rcptto a pointer to a managed string containing the recipient address parameter for the RCPT TO command.

Returns:

-2 if the remote server rejected the command, -1 on general network failure, or 1 on success.

Definition at line 219 of file relay.c.

References client_print(), client_read_line(), log_pedantic, st_char_get(), st_length_get(), and st_length_int().

Referenced by portal_smtp_relay_message(), smtp_bounce(), smtp_forward_message(), smtp_relay_message(), smtp_reply(), and smtp_send_message().

5.119 src/engine/config/relay/relay.h File Reference

Data Structures

- struct [relay_keys_t](#)
- struct [relay_t](#)

Functions

- void [relay_free](#) (void)
Free magma's relay server configuration options.
- [bool_t relay_validate](#) (void)
Validate all of the configured magma relay server instances.
- void [relay_output_settings](#) (void)
- [relay_t * relay_alloc](#) (uint32_t [number](#))
Allocate and initialize a magma server relay object, or return it if it already exists.
- [bool_t relay_config](#) ([stringer_t *name](#), [stringer_t *value](#))
Set the value of a relay server configuration entry by name.
- [bool_t relay_set_value](#) ([relay_keys_t *setting](#), [relay_t *relay](#), [stringer_t *value](#))
Set a specified key for a relay server configuration entry.
- void [relay_output_help](#) (void)
Log all relay key information to be returned via "magma -h".

5.119.1 Function Documentation

5.119.1.1 [relay_t* relay_alloc](#) (uint32_t [number](#))

Allocate and initialize a magma server relay object, or return it if it already exists.

Parameters:

number a zero-based index into the global relay server table where the entry should be created and/or queried.

Returns:

NULL on failure, or a pointer to the requested relay server object on success.

Definition at line 86 of file relay.c.

References [magma_t::host](#), [log_critical](#), [magma](#), [mm_alloc\(\)](#), [magma_t::relay](#), [relay_keys](#), and [relay_set_value\(\)](#).

Referenced by [relay_config\(\)](#).

5.119.1.2 [bool_t relay_config](#) ([stringer_t *name](#), [stringer_t *value](#))

Set the value of a relay server configuration entry by name.

Note:

This function is designed to operate on lines from a configuration source, like a file or database. When a named server is referenced for the first time, a new relay server configuration instance will be allocated.

Parameters:

name a managed string containing the human-readable relay server configuration parameter to be set.

value a managed string containing the new value of the relay server config key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 448 of file relay.c.

References bracket_extract_pl(), CONSTANT, log_critical, MAGMA_RELAY_INSTANCES, NULLER, pl_char_get(), pl_empty(), pl_length_get(), PLACER, placer_t, relay_alloc(), relay_keys, relay_set_value(), st_char_get(), st_cmp_ci_eq(), st_cmp_ci_starts(), st_length_get(), st_length_int(), and uint32_conv_bl().

Referenced by config_load_cmdline_settings(), config_load_database_settings(), and config_load_file_settings().

5.119.1.3 void relay_free (void)

Free magma's relay server configuration options.

Note:

This function assumes all worker threads have finished, and should only be called during the normal shutdown process.

Parameters:

This function returns no value.

Definition at line 39 of file relay.c.

References magma_t::host, log_pedantic, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_ENUM, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_RELAY_INSTANCES, mm_free(), relay_keys_t::norm, ns_empty(), ns_free(), relay_keys_t::offset, magma_t::relay, relay_keys, st_empty, st_free(), type(), and multi_t::type.

Referenced by config_free().

5.119.1.4 void relay_output_help (void)

Log all relay key information to be returned via "magma -h".

Returns:

This function returns no value.

Definition at line 508 of file relay.c.

References multi_t::bl, config_output_value_generic(), log_info, log_options, M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE_DISABLE, M_LOG_TIME_DISABLE, relay_keys_t::norm, relay_keys, relay_keys_t::required, and multi_t::val.

Referenced by config_output_help().

5.119.1.5 void relay_output_settings (void)

Definition at line 228 of file relay.c.

References magma_t::host, log_info, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_RELAY_INSTANCES, relay_keys_t::norm, ns_empty(), relay_keys_t::offset, magma_t::relay, relay_keys, st_char_get(), st_empty, st_length_int(), multi_t::type, and type().

Referenced by config_output_settings().

5.119.1.6 bool_t relay_set_value (relay_keys_t * *setting*, relay_t * *relay*, stringer_t * *value*)

Set a specified key for a relay server configuration entry.

Note:

This function will allocate space for a copy of the key value, and return an error if it is not able to convert it to the proper key data type.

Parameters:

- setting* a pointer to the relay server configuration key to be set.
- relay* a pointer to the relay server configuration entry to be adjusted.
- value* a managed string containing the new value of the config key.

Returns:

true if the value was successfully set, or false on error.

Definition at line 303 of file relay.c.

References multi_t::binary, CONSTANT, CONTIGUOUS, HEAP, multi_t::i32, multi_t::i64, multi_t::i8, int32_conv_st(), int64_conv_st(), int8_conv_st(), log_critical, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, MANAGED_T, relay_keys_t::name, relay_keys_t::norm, multi_t::ns, ns_dupe(), ns_empty(), ns_free(), ns_import(), relay_keys_t::offset, relay_counter(), multi_t::st, st_char_get(), st_cmp_ci_eq(), st_dupe_opts(), st_empty, st_free(), st_length_get(), type(), multi_t::type, multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, uint16_conv_st(), uint32_conv_st(), uint64_conv_st(), uint8_conv_st(), and multi_t::val.

Referenced by relay_alloc(), and relay_config().

5.119.1.7 bool_t relay_validate (void)

Validate all of the configured magma relay server instances. Iterates through all of the server structures and verifies that each was supplied with valid values for all of its required keys. This function also applies the application specific validation rules. Specifically it verifies that only one server structure has been configured to use a given port number.

Note:

This makes set that all required config keys have been set, and that at least one host has been configured.

Returns:

true if all relay servers have been validated, or false on failure.

Definition at line 115 of file relay.c.

References magma_t::host, multi_t::i16, multi_t::i32, multi_t::i64, multi_t::i8, log_critical, log_pedantic, M_TYPE_BOOLEAN, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_RELAY_INSTANCES, relay_keys_t::norm, ns_empty(),

relay_keys_t::offset, magma_t::relay, relay_keys, st_empty, multi_t::type, type(), multi_t::u16, multi_t::u32, multi_t::u64, multi_t::u8, and multi_t::val.

Referenced by config_validate_settings().

5.120 src/engine/config/servers/servers.c File Reference

```
#include "magma.h"
#include "keys.h"
```

Functions

- `uint64_t servers_get_count_using_port (uint32_t port)`
Get the number of servers that are configured to use a specified port.
- `server_t * servers_get_by_socket (int sockd)`
Lookup the server instance associated with a socket descriptor.
- `server_t * servers_get_by_protocol (uint32_t protocol, bool_t tls)`
Lookup the server instance associated with a particular protocol type.
- `bool_t servers_network_start (void)`
Setup the listening sockets for all configured servers, or shut them all down if any one of them fails.
- `bool_t servers_encryption_start (void)`
Create a TLS context for each TLS enabled server instance.
- `void servers_network_stop (void)`
Close the listening sockets of all running servers.
- `void servers_encryption_stop (void)`
Destroy the transport security context for each TLS enabled servers.
- `void servers_free (void)`
Free magma's server configuration options.
- `server_t * servers_alloc (uint32_t number)`
Allocate a new magma server configuration entry and initialize it to the default values.
- `bool_t servers_validate (void)`
Validate all of the configured magmad server instances.
- `void servers_output_settings (void)`
Log the full contents of the magma server instance settings.
- `bool_t servers_set_value (server_keys_t *setting, server_t *server, stringer_t *value)`
Set the value of a key for a specified magma server configuration entry.
- `bool_t servers_config (stringer_t *name, stringer_t *value)`
Set the value of a magma server configuration entry by name.
- `void servers_output_help (void)`
Log all server key information to be returned via the "magmad -h|--help" command.

5.120.1 Function Documentation

5.120.1.1 `server_t* servers_alloc (uint32_t number)`

Allocate a new magma server configuration entry and initialize it to the default values. [servers.c](#)

Parameters:

number the index of the magma server instance in the global magma server array.

Returns:

NULL on failure, or a pointer to the requested magma server configuration entry on success.

Definition at line 193 of file `servers.c`.

References `log_critical`, `magma`, `mm_alloc()`, `server_t::network`, `server_keys`, `magma_t::servers`, `servers_set_value()`, and `server_t::sockd`.

Referenced by `servers_config()`.

5.120.1.2 `bool_t servers_config (stringer_t * name, stringer_t * value)`

Set the value of a magma server configuration entry by name.

Note:

This function is designed to operate on lines from a configuration source, like a file or database. When a named server is referenced for the first time, a new magma server configuration instance will be allocated.

Parameters:

name a managed string containing the human-readable magma server configuration parameter to be set.

value a managed string containing the new value of the magma server config key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 666 of file `servers.c`.

References `bracket_extract_pl()`, `CONSTANT`, `log_critical`, `MAGMA_SERVER_INSTANCES`, `NULLER`, `pl_char_get()`, `pl_empty()`, `pl_length_get()`, `PLACER`, `placer_t`, `server_keys`, `servers_alloc()`, `servers_set_value()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_cmp_ci_starts()`, `st_length_get()`, `st_length_int()`, and `uint32_conv_bl()`.

Referenced by `config_load_cmdline_settings()`, `config_load_database_settings()`, and `config_load_file_settings()`.

5.120.1.3 `bool_t servers_encryption_start (void)`

Create a TLS context for each TLS enabled server instance.

Returns:

true on success or false on failure.

Definition at line 101 of file `servers.c`.

References `server_t::certificate`, `DMTP`, `server_t::enabled`, `magma`, `MAGMA_SERVER_INSTANCES`, `server_t::protocol`, `magma_t::servers`, `server_t::tls`, and `tls_server_create()`.

Referenced by `process_start()`.

5.120.1.4 void servers_encryption_stop (void)

Destroy the transport security context for each TLS enabled servers.

Returns:

This function returns no value.

Definition at line 132 of file servers.c.

References server_t::context, server_t::enabled, magma, MAGMA_SERVER_INSTANCES, magma_t::servers, server_t::tls, and tls_server_destroy().

Referenced by process_stop().

5.120.1.5 void servers_free (void)

Free magma's server configuration options.

Note:

This function assumes all worker threads have finished, and should only be called during the normal shutdown process.

Parameters:

This function returns no value.

Definition at line 146 of file servers.c.

References log_pedantic, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_ENUM, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_SERVER_INSTANCES, mm_free(), server_keys_t::norm, ns_empty(), ns_free(), server_keys_t::offset, server_keys, magma_t::servers, st_empty, st_free(), type(), and multi_t::type.

Referenced by config_free().

5.120.1.6 server_t* servers_get_by_protocol (uint32_t protocol, bool_t tls)

Lookup the server instance associated with a particular protocol type.

Note:

This function doesn't allow you to specify which server instance to return, if more than one is available.

Parameters:

protocol the enumerated protocol type being looked up.

tls whether the server instance should be a TCP server, or a TLS server instance.

Returns:

NULL if no server instances are configured, or a pointer to the server structure for a protocol.

Definition at line 54 of file servers.c.

References DMTP, server_t::enabled, HTTP, IMAP, log_pedantic, magma, MAGMA_SERVER_INSTANCES, MOLTEN, server_t::network, POP, server_t::protocol, magma_t::servers, SMTP, SUBMISSION, TCP_PORT, TLS_PORT, server_t::type, and type().

5.120.1.7 `server_t* servers_get_by_socket (int sockd)`

Lookup the server instance associated with a socket descriptor.

Parameters:

sockd the socket file descriptor to be looked up.

Returns:

NULL on failure, or a pointer to the server structure for the specified file descriptor on success.

Definition at line 34 of file servers.c.

References magma, MAGMA_SERVER_INSTANCES, server_t::network, magma_t::servers, and server_t::sockd.

5.120.1.8 `uint64_t servers_get_count_using_port (uint32_t port)`

Get the number of servers that are configured to use a specified port.

Parameters:

port the port number to be matched against the server list.

Returns:

the number of servers configured to use the port.

Definition at line 17 of file servers.c.

References count, magma, MAGMA_SERVER_INSTANCES, server_t::network, server_t::port, and magma_t::servers.

Referenced by net_trigger(), and servers_validate().

5.120.1.9 `bool_t servers_network_start (void)`

Setup the listening sockets for all configured servers, or shut them all down if any one of them fails.

Note:

If any of the servers have an empty name or domain, the value of magma.host.name will be used instead.

Returns:

true if all server listening sockets were initialized successfully, or false on failure.

Definition at line 80 of file servers.c.

References server_t::domain, server_t::enabled, magma_t::host, magma, MAGMA_SERVER_INSTANCES, magma_t::name, server_t::name, net_init(), ns_length_get(), magma_t::servers, servers_network_stop(), st_empty, and st_import().

Referenced by process_start().

5.120.1.10 `void servers_network_stop (void)`

Close the listening sockets of all running servers.

Returns:

This function returns no value.

Definition at line 119 of file servers.c.

References `server_t::enabled`, `magma`, `MAGMA_SERVER_INSTANCES`, `net_shutdown()`, `server_t::network`, `magma_t::servers`, and `server_t::sockd`.

Referenced by `process_stop()`, and `servers_network_start()`.

5.120.1.11 void servers_output_help (void)

Log all server key information to be returned via the "magmad -h|--help" command.

Returns:

This function returns no value.

Definition at line 723 of file servers.c.

References `multi_t::bl`, `config_output_value_generic()`, `log_info`, `log_options`, `M_LOG_FILE_DISABLE`, `M_LOG_FUNCTION_DISABLE`, `M_LOG_LINE_DISABLE`, `M_LOG_LINE_FEED_DISABLE`, `M_LOG_STACK_TRACE_DISABLE`, `M_LOG_TIME_DISABLE`, `server_keys_t::norm`, `server_keys_t::required`, `server_keys`, and `multi_t::val`.

Referenced by `config_output_help()`.

5.120.1.12 void servers_output_settings (void)

Log the full contents of the magma server instance settings.

Returns:

This function returns no value.

Definition at line 368 of file servers.c.

References `CONSTANT`, `DMTP`, `GENERIC`, `HTTP`, `IMAP`, `log_info`, `log_pedantic`, `M_TYPE_BOOLEAN`, `M_TYPE_ENUM`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `magma`, `MAGMA_SERVER_INSTANCES`, `MOLTEN`, `server_keys_t::norm`, `ns_empty()`, `NULLER`, `server_keys_t::offset`, `POP`, `server_keys`, `magma_t::servers`, `SMTP`, `st_char_get()`, `st_cmp_cs_eq()`, `st_empty`, `st_length_int()`, `SUBMISSION`, `TCP_PORT`, `TLS_PORT`, `multi_t::type`, and `type()`.

Referenced by `config_output_settings()`.

5.120.1.13 bool_t servers_set_value (server_keys_t * setting, server_t * server, stringer_t * value)

Set the value of a key for a specified magma server configuration entry.

Note:

This function will allocate space for a copy of the key value, and return an error if it is not able to convert it to the proper key data type.

Parameters:

setting a pointer to the magma server key to be set.

server a pointer to the magma server configuration entry to be modified.

value a managed string containing the new value of the key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 474 of file servers.c.

References `multi_t::binary`, `CONSTANT`, `CONTIGUOUS`, `DMTP`, `HEAP`, `HTTP`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `IMAP`, `int32_conv_st()`, `int64_conv_st()`, `int8_conv_st()`, `log_critical`, `M_TYPE_BOOLEAN`, `M_TYPE_ENUM`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `MANAGED_T`, `MOLTEN`, `server_keys_t::name`, `server_keys_t::norm`, `multi_t::ns`, `ns_dupe()`, `ns_empty()`, `ns_free()`, `ns_import()`, `NULLER`, `server_keys_t::offset`, `POP`, `SMTP`, `multi_t::st`, `st_char_get()`, `st_cmp_ci_eq()`, `st_dupe_opts()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_int()`, `SUBMISSION`, `TCP_PORT`, `TLS_PORT`, `type()`, `multi_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, `uint16_conv_st()`, `uint32_conv_st()`, `uint64_conv_st()`, `uint8_conv_st()`, and `multi_t::val`.

Referenced by `servers_alloc()`, and `servers_config()`.

5.120.1.14 `bool_t servers_validate (void)`

Validate all of the configured magmad server instances.

Note:

This function also checks to see that no servers are binding to the same port.

Returns:

true if all servers have been validated, or false on failure.

Definition at line 220 of file servers.c.

References `buflen`, `bufptr`, `server_t::certificate`, `CONSTANT`, `DMTP`, `file_readwritable()`, `file_world_accessible()`, `GENERIC`, `multi_t::i16`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `log_critical`, `log_pedantic`, `M_TYPE_BOOLEAN`, `M_TYPE_ENUM`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `magma`, `MAGMA_SERVER_INSTANCES`, `server_t::network`, `server_keys_t::norm`, `ns_empty()`, `NULLER`, `server_keys_t::offset`, `server_t::port`, `server_t::protocol`, `server_keys`, `magma_t::servers`, `servers_get_count_using_port()`, `st_cmp_ci_eq()`, `st_empty`, `server_t::tls`, `TLS_PORT`, `multi_t::type`, `type()`, `server_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, and `multi_t::val`.

Referenced by `config_validate_settings()`.

5.121 src/engine/config/servers/servers.h File Reference

Data Structures

- struct [server_keys_t](#)
- struct [server_t](#)
- struct [server_config_t](#)

Enumerations

- enum [M_PORT](#) { [TCP_PORT](#) = 1, [TLS_PORT](#) }
- enum [M_PROTOCOL](#) {
[GENERIC](#) = 0, [MOLTEN](#) = 1, [HTTP](#), [POP](#),
[IMAP](#), [SMTP](#), [DMTP](#), [SUBMISSION](#) }

Functions

- [server_t * servers_alloc](#) (uint32_t number)
[servers.c](#)
- [bool_t servers_config](#) (stringer_t *name, stringer_t *value)
Set the value of a magma server configuration entry by name.
- [bool_t servers_encryption_start](#) (void)
Create a TLS context for each TLS enabled server instance.
- [void servers_encryption_stop](#) (void)
Destroy the transport security context for each TLS enabled servers.
- [void servers_free](#) (void)
Free magma's server configuration options.
- [server_t * servers_get_by_protocol](#) (uint32_t protocol, bool_t tls)
Lookup the server instance associated with a particular protocol type.
- [server_t * servers_get_by_socket](#) (int sockd)
Lookup the server instance associated with a socket descriptor.
- [uint64_t servers_get_count_using_port](#) (uint32_t port)
Get the number of servers that are configured to use a specified port.
- [bool_t servers_network_start](#) (void)
Setup the listening sockets for all configured servers, or shut them all down if any one of them fails.
- [void servers_network_stop](#) (void)
Close the listening sockets of all running servers.
- [void servers_output_help](#) (void)
Log all server key information to be returned via the "magmad -h|--help" command.
- [void servers_output_settings](#) (void)
Log the full contents of the magma server instance settings.

- [bool_t servers_set_value](#) ([server_keys_t](#) *setting, [server_t](#) *server, [stringer_t](#) *value)

Set the value of a key for a specified magma server configuration entry.

- [bool_t servers_validate](#) (void)

Validate all of the configured magmad server instances.

5.121.1 Enumeration Type Documentation

5.121.1.1 enum M_PORT

Enumerator:

TCP_PORT

TLS_PORT

Definition at line 11 of file servers.h.

5.121.1.2 enum M_PROTOCOL

Enumerator:

GENERIC

MOLTEN

HTTP

POP

IMAP

SMTP

DMTP

SUBMISSION

Definition at line 16 of file servers.h.

5.121.2 Function Documentation

5.121.2.1 server_t* servers_alloc (uint32_t number)

[servers.c](#) [servers.c](#)

Parameters:

number the index of the magma server instance in the global magma server array.

Returns:

NULL on failure, or a pointer to the requested magma server configuration entry on success.

Definition at line 193 of file servers.c.

References [log_critical](#), [magma](#), [mm_alloc\(\)](#), [server_t::network](#), [server_keys](#), [magma_t::servers](#), [servers_set_value\(\)](#), and [server_t::sockd](#).

Referenced by [servers_config\(\)](#).

5.121.2.2 **bool_t servers_config (stringer_t * name, stringer_t * value)**

Set the value of a magma server configuration entry by name.

Note:

This function is designed to operate on lines from a configuration source, like a file or database. When a named server is referenced for the first time, a new magma server configuration instance will be allocated.

Parameters:

name a managed string containing the human-readable magma server configuration parameter to be set.

value a managed string containing the new value of the magma server config key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 666 of file servers.c.

References bracket_extract_pl(), CONSTANT, log_critical, MAGMA_SERVER_INSTANCES, NULLER, pl_char_get(), pl_empty(), pl_length_get(), PLACER, placer_t, server_keys, servers_alloc(), servers_set_value(), st_char_get(), st_cmp_ci_eq(), st_cmp_ci_starts(), st_length_get(), st_length_int(), and uint32_conv_bl().

Referenced by config_load_cmdline_settings(), config_load_database_settings(), and config_load_file_settings().

5.121.2.3 **bool_t servers_encryption_start (void)**

Create a TLS context for each TLS enabled server instance.

Returns:

true on success or false on failure.

Definition at line 101 of file servers.c.

References server_t::certificate, DMTP, server_t::enabled, magma, MAGMA_SERVER_INSTANCES, server_t::protocol, magma_t::servers, server_t::tls, and tls_server_create().

Referenced by process_start().

5.121.2.4 **void servers_encryption_stop (void)**

Destroy the transport security context for each TLS enabled servers.

Returns:

This function returns no value.

Definition at line 132 of file servers.c.

References server_t::context, server_t::enabled, magma, MAGMA_SERVER_INSTANCES, magma_t::servers, server_t::tls, and tls_server_destroy().

Referenced by process_stop().

5.121.2.5 **void servers_free (void)**

Free magma's server configuration options.

Note:

This function assumes all worker threads have finished, and should only be called during the normal shutdown process.

Parameters:

This function returns no value.

Definition at line 146 of file servers.c.

References log_pedantic, M_TYPE_BLOCK, M_TYPE_BOOLEAN, M_TYPE_DOUBLE, M_TYPE_ENUM, M_TYPE_FLOAT, M_TYPE_INT16, M_TYPE_INT32, M_TYPE_INT64, M_TYPE_INT8, M_TYPE_NULLER, M_TYPE_STRINGER, M_TYPE_UINT16, M_TYPE_UINT32, M_TYPE_UINT64, M_TYPE_UINT8, magma, MAGMA_SERVER_INSTANCES, mm_free(), server_keys_t::norm, ns_empty(), ns_free(), server_keys_t::offset, server_keys, magma_t::servers, st_empty, st_free(), type(), and multi_t::type.

Referenced by config_free().

5.121.2.6 server_t* servers_get_by_protocol (uint32_t protocol, bool_t tls)

Lookup the server instance associated with a particular protocol type.

Note:

This function doesn't allow you to specify which server instance to return, if more than one is available.

Parameters:

protocol the enumerated protocol type being looked up.

tls whether the server instance should be a TCP server, or a TLS server instance.

Returns:

NULL if no server instances are configured, or a pointer to the server structure for a protocol.

Definition at line 54 of file servers.c.

References DMTP, server_t::enabled, HTTP, IMAP, log_pedantic, magma, MAGMA_SERVER_INSTANCES, MOLTEN, server_t::network, POP, server_t::protocol, magma_t::servers, SMTP, SUBMISSION, TCP_PORT, TLS_PORT, server_t::type, and type().

5.121.2.7 server_t* servers_get_by_socket (int sockd)

Lookup the server instance associated with a socket descriptor.

Parameters:

sockd the socket file descriptor to be looked up.

Returns:

NULL on failure, or a pointer to the server structure for the specified file descriptor on success.

Definition at line 34 of file servers.c.

References magma, MAGMA_SERVER_INSTANCES, server_t::network, magma_t::servers, and server_t::sockd.

5.121.2.8 uint64_t servers_get_count_using_port (uint32_t port)

Get the number of servers that are configured to use a specified port.

Parameters:

port the port number to be matched against the server list.

Returns:

the number of servers configured to use the port.

Definition at line 17 of file servers.c.

References count, magma, MAGMA_SERVER_INSTANCES, server_t::network, server_t::port, and magma_t::servers.

Referenced by net_trigger(), and servers_validate().

5.121.2.9 bool_t servers_network_start (void)

Setup the listening sockets for all configured servers, or shut them all down if any one of them fails.

Note:

If any of the servers have an empty name or domain, the value of magma.host.name will be used instead.

Returns:

true if all server listening sockets were initialized successfully, or false on failure.

Definition at line 80 of file servers.c.

References server_t::domain, server_t::enabled, magma_t::host, magma, MAGMA_SERVER_INSTANCES, magma_t::name, server_t::name, net_init(), ns_length_get(), magma_t::servers, servers_network_stop(), st_empty, and st_import().

Referenced by process_start().

5.121.2.10 void servers_network_stop (void)

Close the listening sockets of all running servers.

Returns:

This function returns no value.

Definition at line 119 of file servers.c.

References server_t::enabled, magma, MAGMA_SERVER_INSTANCES, net_shutdown(), server_t::network, magma_t::servers, and server_t::sockd.

Referenced by process_stop(), and servers_network_start().

5.121.2.11 void servers_output_help (void)

Log all server key information to be returned via the "magmad -h|--help" command.

Returns:

This function returns no value.

Definition at line 723 of file servers.c.

References multi_t::bl, config_output_value_generic(), log_info, log_options, M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE_DISABLE, M_LOG_TIME_DISABLE, server_keys_t::norm, server_keys_t::required, server_keys, and multi_t::val.

Referenced by config_output_help().

5.121.2.12 void servers_output_settings (void)

Log the full contents of the magma server instance settings.

Returns:

This function returns no value.

Definition at line 368 of file servers.c.

References `CONSTANT`, `DMTP`, `GENERIC`, `HTTP`, `IMAP`, `log_info`, `log_pedantic`, `M_TYPE_BOOLEAN`, `M_TYPE_ENUM`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `magma`, `MAGMA_SERVER_INSTANCES`, `MOLTEN`, `server_keys_t::norm`, `ns_empty()`, `NULLER`, `server_keys_t::offset`, `POP`, `server_keys`, `magma_t::servers`, `SMTP`, `st_char_get()`, `st_cmp_cs_eq()`, `st_empty`, `st_length_int()`, `SUBMISSION`, `TCP_PORT`, `TLS_PORT`, `multi_t::type`, and `type()`.

Referenced by `config_output_settings()`.

5.121.2.13 bool_t servers_set_value (server_keys_t * setting, server_t * server, stringer_t * value)

Set the value of a key for a specified magma server configuration entry.

Note:

This function will allocate space for a copy of the key value, and return an error if it is not able to convert it to the proper key data type.

Parameters:

setting a pointer to the magma server key to be set.

server a pointer to the magma server configuration entry to be modified.

value a managed string containing the new value of the key.

Returns:

true if the value was successfully set, or false on failure.

Definition at line 474 of file servers.c.

References `multi_t::binary`, `CONSTANT`, `CONTIGUOUS`, `DMTP`, `HEAP`, `HTTP`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `IMAP`, `int32_conv_st()`, `int64_conv_st()`, `int8_conv_st()`, `log_critical`, `M_TYPE_BOOLEAN`, `M_TYPE_ENUM`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `MANAGED_T`, `MOLTEN`, `server_keys_t::name`, `server_keys_t::norm`, `multi_t::ns`, `ns_dupe()`, `ns_empty()`, `ns_free()`, `ns_import()`, `NULLER`, `server_keys_t::offset`, `POP`, `SMTP`, `multi_t::st`, `st_char_get()`, `st_cmp_ci_eq()`, `st_dupe_opts()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_int()`, `SUBMISSION`, `TCP_PORT`, `TLS_PORT`, `type()`, `multi_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, `uint16_conv_st()`, `uint32_conv_st()`, `uint64_conv_st()`, `uint8_conv_st()`, and `multi_t::val`.

Referenced by `servers_alloc()`, and `servers_config()`.

5.121.2.14 bool_t servers_validate (void)

Validate all of the configured magmad server instances.

Note:

This function also checks to see that no servers are binding to the same port.

Returns:

true if all servers have been validated, or false on failure.

Definition at line 220 of file servers.c.

References `buflen`, `bufptr`, `server_t::certificate`, `CONSTANT`, `DMTP`, `file_readwritable()`, `file_world_accessible()`, `GENERIC`, `multi_t::i16`, `multi_t::i32`, `multi_t::i64`, `multi_t::i8`, `log_critical`, `log_pedantic`, `M_TYPE_BOOLEAN`, `M_TYPE_ENUM`, `M_TYPE_INT16`, `M_TYPE_INT32`, `M_TYPE_INT64`, `M_TYPE_INT8`, `M_TYPE_NULLER`, `M_TYPE_STRINGER`, `M_TYPE_UINT16`, `M_TYPE_UINT32`, `M_TYPE_UINT64`, `M_TYPE_UINT8`, `magma`, `MAGMA_SERVER_INSTANCES`, `server_t::network`, `server_keys_t::norm`, `ns_empty()`, `NULLER`, `server_keys_t::offset`, `server_t::port`, `server_t::protocol`, `server_keys`, `magma_t::servers`, `servers_get_count_using_port()`, `st_cmp_ci_eq()`, `st_empty`, `server_t::tls`, `TLS_PORT`, `multi_t::type`, `type()`, `server_t::type`, `multi_t::u16`, `multi_t::u32`, `multi_t::u64`, `multi_t::u8`, and `multi_t::val`.

Referenced by `config_validate_settings()`.

5.122 src/servers/servers.h File Reference

```
#include "http/http.h"
#include "imap/imap.h"
#include "molten/molten.h"
#include "pop/pop.h"
#include "smtp/smtp.h"
#include "dmtip/dmtip.h"
```

Defines

- #define [M_SSL_BIO_NOCLOSE](#) 0x00

5.122.1 Define Documentation

5.122.1.1 #define M_SSL_BIO_NOCLOSE 0x00

Definition at line 11 of file servers.h.

Referenced by `imap_starttls()`, `pop_starttls()`, `protocol_secure()`, and `smtp_starttls()`.

5.123 src/engine/context/args.c File Reference

```
#include "magma.h"
```

Functions

- void [display_usage](#) (void)
Display the magma usage info if a user requests help or invokes it with incorrect options.
- [bool_t args_parse](#) (int argc, char *argv[])
Process any command line arguments supplied to magma.

Variables

- [stringer_t * cmdline_config_data](#)
- [bool_t exit_and_dump](#)

5.123.1 Function Documentation

5.123.1.1 [bool_t args_parse](#) (int *argc*, char * *argv*[])

Process any command line arguments supplied to magma.

Note:

A few command line options are supported: -h (--help), -d (--dump), -v (--version), and -c (--config). Otherwise, the magma config file path will be loaded from the command line. If not an option starting with "-", the final command line argument is assumed to be the path of the magma configuration file.

Returns:

true if the command line arguments were parsed successfully, or false if there was an error.

Definition at line 39 of file args.c.

References [build_commit\(\)](#), [build_stamp\(\)](#), [build_version\(\)](#), [cmdline_config_data](#), [magma_t::config](#), [config_load_defaults\(\)](#), [config_output_help\(\)](#), [display_usage\(\)](#), [exit_and_dump](#), [magma_t::file](#), [HEAP](#), [JOINTED](#), [log_critical](#), [log_info](#), [magma](#), [MAGMA_FILEPATH_MAX](#), [MANAGED_T](#), [mm_cmp_cs_eq\(\)](#), [NULLER](#), [PLACER](#), [st_append](#), [st_cmp_cs_eq\(\)](#), and [st_dupe_opts\(\)](#).

Referenced by [main\(\)](#).

5.123.1.2 void [display_usage](#) (void)

Display the magma usage info if a user requests help or invokes it with incorrect options. [args.c](#)

Returns:

This function returns no value.

Definition at line 19 of file args.c.

References [log_info](#).

Referenced by [args_parse\(\)](#).

5.123.2 Variable Documentation

5.123.2.1 stringer_t* cmdline_config_data

Definition at line 14 of file global.c.

Referenced by args_parse(), and config_load_cmdline_settings().

5.123.2.2 bool_t exit_and_dump

Definition at line 13 of file global.c.

5.124 src/engine/context/context.h File Reference

Functions

- void `display_usage` (void)
args.c
- `bool_t` `args_parse` (int argc, char *argv[])
Process any command line arguments supplied to magma.
- `bool_t` `sanity_check` (void)
Perform a series of system-wide sanity checks at process startup.
- void `process_stop` (void)
Execute the magma shutdown routines.
- `bool_t` `process_start` (void)
Execute the magma initializations routines, making sure they all run properly.
- `bool_t` `signal_start` (void)
signal.c
- `bool_t` `signal_thread_start` (void)
Bind a SIGALRM handler for the calling thread.
- void `signal_segfault` (int signal)
Handle signals that indicate the program was killed because of an invalid operation (including SIGSEGV).
- void `signal_shutdown` (int signal)
A function to handle receipt of shutdown signals and allow for graceful exits.
- void `signal_status` (int signal)
A generic worker thread signal handler entry point.
- `bool_t` `system_change_root_directory` (void)
system.c
- `bool_t` `system_fork_daemon` (void)
Daemonize into the background, if the magma.system.daemonize config option is set.
- `bool_t` `system_init_core_dumps` (void)
Set the magma core dump size rlimit, if magma.system.enable_core_dumps was enabled.
- `bool_t` `system_init_impersonation` (void)
Set process privileges to run as a specified user if magma.system.impersonate_user is set.
- `bool_t` `system_init_resource_limits` (void)
Increase process resource limits, if magma.system.increase_resource_limits is set.
- `bool_t` `system_init_umask` (void)
Set the process umask for new file/dir creation to O_RDWR.
- `uint64_t` `system_ulimit_cur` (`int_t` resource)

Get the soft system limit for a specified resource.

- `uint64_t system_ulimit_max (int_t resource)`

Get the hard system limit for a specified resource.

- `bool_t thread_start (void)`

thread.c

- `void thread_stop (void)`

Prepare a thread to exit by destroying its MySQL and OpenSSL thread storage, and the thread specific mail cache.

5.124.1 Function Documentation

5.124.1.1 `bool_t args_parse (int argc, char * argv[])`

Process any command line arguments supplied to magma.

Note:

A few command line options are supported: `-h` (`--help`), `-d` (`--dump`), `-v` (`--version`), and `-c` (`--config`). Otherwise, the magma config file path will be loaded from the command line. If not an option starting with "-", the final command line argument is assumed to be the path of the magma configuration file.

Returns:

true if the command line arguments were parsed successfully, or false if there was an error.

Definition at line 39 of file `args.c`.

References `build_commit()`, `build_stamp()`, `build_version()`, `cmdline_config_data`, `magma_t::config`, `config_load_defaults()`, `config_output_help()`, `display_usage()`, `exit_and_dump`, `magma_t::file`, `HEAP`, `JOINTED`, `log_critical`, `log_info`, `magma`, `MAGMA_FILEPATH_MAX`, `MANAGED_T`, `mm_cmp_cs_eq()`, `NULLER`, `PLACER`, `st_append`, `st_cmp_cs_eq()`, and `st_dupe_opts()`.

Referenced by `main()`.

5.124.1.2 `void display_usage (void)`

`args.c` `args.c`

Returns:

This function returns no value.

Definition at line 19 of file `args.c`.

References `log_info`.

Referenced by `args_parse()`.

5.124.1.3 `bool_t process_start (void)`

Execute the magma initializations routines, making sure they all run properly.

Note:

If any of the init routines returns with failure, this function fails and logs an error message. Certain checks are made that the init routines are run in a particular order, especially in waiting for daemonization and privilege dropping before starting logging. The system maintenance thread is also launched from here.

Returns:

true if all starter functions returned true, or false if any of them failed..

Definition at line 163 of file process.c.

References `cache_start()`, `config_load_cmdline_settings()`, `config_load_database_settings()`, `config_load_defaults()`, `config_load_file_settings()`, `config_validate_settings()`, `deprecated_ecies_start()`, `dkim_start()`, `dspam_start()`, `exit_and_dump`, `magma_t::host`, `http_content_start()`, `magma_t::init`, `lib_load()`, `log_critical`, `log_options`, `log_start()`, `M_LOG_FILE_DISABLE`, `M_LOG_FUNCTION_DISABLE`, `M_LOG_INFO`, `M_LOG_LINE_DISABLE`, `M_LOG_STACK_TRACE_DISABLE`, `M_LOG_TIME_DISABLE`, `magma`, `MAGMA_HOSTNAME_MAX`, `mail_cache_start()`, `maint`, `mm_alloc()`, `mm_free()`, `mm_sec_start()`, `magma_t::name`, `obj_cache_start()`, `magma_t::page_length`, `prime_start()`, `process_maint()`, `protocol_init()`, `queue_init()`, `rand_start()`, `sanity_check()`, `servers_encryption_start()`, `servers_network_start()`, `signal_start()`, `spf_start()`, `spool_start()`, `sql_start()`, `ssl_start()`, `stats_init()`, `status_process()`, `system_change_root_directory()`, `system_fork_daemon()`, `system_init_core_dumps()`, `system_init_impersonation()`, `system_init_resource_limits()`, `system_init_umask()`, `tank_start()`, `thread_launch()`, `virus_start()`, `warehouse_start()`, and `xml_start()`.

Referenced by `main()`.

5.124.1.4 void process_stop (void)

Execute the magma shutdown routines.

Note:

This function will execute a shutdown routine for each of the startup initialization routines that was performed. It will also signal and terminate the maintenance thread.

Returns:

This function returns no value.

Definition at line 66 of file process.c.

References `cache_stop()`, `config_free()`, `deprecated_ecies_stop()`, `dkim_stop()`, `dspam_stop()`, `http_content_stop()`, `magma_t::init`, `lib_unload()`, `log_critical`, `log_info`, `log_options`, `log_pedantic`, `M_LOG_FILE_DISABLE`, `M_LOG_FUNCTION_DISABLE`, `M_LOG_INFO`, `M_LOG_LINE_DISABLE`, `M_LOG_STACK_TRACE_DISABLE`, `M_LOG_TIME_DISABLE`, `magma`, `mail_cache_stop()`, `maint`, `mm_free()`, `mm_sec_stop()`, `obj_cache_stop()`, `prime_stop()`, `queue_shutdown()`, `rand_stop()`, `servers_encryption_stop()`, `servers_network_stop()`, `spf_stop()`, `spool_stop()`, `sql_stop()`, `ssl_stop()`, `stats_shutdown()`, `status`, `tank_stop()`, `thread_join()`, `thread_signal()`, `virus_stop()`, `warehouse_stop()`, and `xml_stop()`.

Referenced by `main()`.

5.124.1.5 bool_t sanity_check (void)

Perform a series of system-wide sanity checks at process startup.

Note:

This function ensures that the standard data types are their expected data sizes, and performs other checks, such as data primitive maximum values.

Returns:

true if the system environment passed all checks and should function correctly, or false if any checks failed.

Definition at line 15 of file sanity.c.

References `CONSTANT`, `MAGMA_CORE_POOL_OBJECTS_LIMIT`, `MAGMA_CORE_POOL_TIMEOUT_LIMIT`, `NULLER`, `placer_t`, `queries`, `record_t`, `SELECT_DOMAINS`, `st_alloc()`, `st_avail_get()`, `st_cmp_cs_eq()`, `st_data_get()`, and `st_free()`.

Referenced by `process_start()`.

5.124.1.6 void signal_segfault (int *signal*)

Handle signals that indicate the program was killed because of an invalid operation (including SIGSEGV).

Note:

The signals handled by this function are: SIGSEGV, SIGFPE, SIGBUS, SIGSYS, and SIGABRT. The handler will respond by logging a stack backtrace, and terminating the program with abort().

Parameters:

signal the number of the signal that was received.

Returns:

This function returns no value.

Definition at line 19 of file signal.c.

References log_options, M_LOG_CRITICAL, M_LOG_STACK_TRACE, mm_wipe(), and signal_name().

Referenced by signal_start().

5.124.1.7 void signal_shutdown (int *signal*)

A function to handle receipt of shutdown signals and allow for graceful exits.

Note:

This function handles the following signals: SIGINT, SIGQUIT, and SIGTERM. The shutdown procedure is as follows: 1. Set status to -1 and sleep for .1 second to allow for normal daemon termination. 2. Signal all worker threads to wake up blocked threads, and sleep one more second. 3. Loop through all possible file descriptors and if the descriptor matches a socket that is bound to a magma server, close it.

Parameters:

signal the number of the signal delivered to the process.

Returns:

This function returns no value.

Definition at line 52 of file signal.c.

References log_critical, MEMORYBUF, mm_wipe(), net_trigger(), queue_signal(), signal_name(), status_signal(), thread_join(), and thread_launch().

Referenced by signal_start().

5.124.1.8 bool_t signal_start (void)

[signal.c](#) [signal.c](#)

Note:

The following handlers are established: signal_shutdown: SIGINT, SIGQUIT, SIGTERM, SIGHUP signal_segfault: SIGSEGV, SIGFPE, SIGBUS, SIGSYS signal_refresh: SIGHUP [ignored]: SIGPIPE

Returns:

true if all signal handlers were successfully registered, or false on failure.

Definition at line 154 of file signal.c.

References log_info, mm_wipe(), signal_refresh(), signal_segfault(), and signal_shutdown().

Referenced by process_start().

5.124.1.9 void signal_status (int *signal*)

A generic worker thread signal handler entry point.

Note:

If the target thread is not in the process of shutting down, display the name of the caught signal.

Parameters:

signal the number of the signal caught by the signal handler.

Returns:

This function returns no value.

Definition at line 93 of file signal.c.

References log_info, signal_name(), and status.

Referenced by signal_thread_start().

5.124.1.10 bool_t signal_thread_start (void)

Bind a SIGALRM handler for the calling thread.

See also:

[signal_status\(\)](#)

Returns:

false on failure or true on success.

TODO: Disabling this signal handler to see if it fixes the shutdown issue.

Definition at line 128 of file signal.c.

References log_info, mm_wipe(), and signal_status().

Referenced by thread_start().

5.124.1.11 bool_t system_change_root_directory (void)

[system.c](#) [system.c](#)

Returns:

true on success or false on failure.

Definition at line 66 of file system.c.

References buflen, bufptr, log_info, magma, magma_t::root_directory, and magma_t::system.

Referenced by process_start().

5.124.1.12 `bool_t system_fork_daemon (void)`

Daemonize into the background, if the `magma.system.daemonize` config option is set.

Returns:

true inside the child process, or false inside the parent process or if an error occurs.

Definition at line 81 of file `system.c`.

References `magma_t::daemonize`, `log_critical`, `log_info`, `magma`, `pid`, `status_process()`, and `magma_t::system`.

Referenced by `process_start()`.

5.124.1.13 `bool_t system_init_core_dumps (void)`

Set the magma core dump size `rlimit`, if `magma.system.enable_core_dumps` was enabled.

Returns:

true if core dumps were successfully enabled or false on failure.

Definition at line 149 of file `system.c`.

References `buflen`, `bufptr`, `magma_t::core_dump_size_limit`, `magma_t::enable_core_dumps`, `log_info`, `magma`, and `magma_t::system`.

Referenced by `process_start()`.

5.124.1.14 `bool_t system_init_impersonation (void)`

Set process privileges to run as a specified user if `magma.system.impersonate_user` is set.

Note:

This function will set the user id and group id to the specified user, and `chdir()` to their home directory.

Returns:

true on success or if `magma.system.impersonate_user` is not set; false on failure to change privileges.

Definition at line 169 of file `system.c`.

References `buflen`, `bufptr`, `magma_t::impersonate_user`, `log_critical`, `log_info`, `magma`, `mm_alloc()`, `mm_free()`, `mm_wipe()`, `status_process()`, and `magma_t::system`.

Referenced by `process_start()`.

5.124.1.15 `bool_t system_init_resource_limits (void)`

Increase process resource limits, if `magma.system.increase_resource_limits` is set.

Note:

Resource limits will be maximized for magma's virtual address space, data and stack segments, available file descriptors and sub-processes, and allowed file sizes. If `output_resource_limits` is enabled, the state of the process resource limits will be dumped to the log afterwards.

Returns:

This function always returns true (errors are logged).

Definition at line 244 of file system.c.

References magma_t::config, magma_t::increase_resource_limits, log_info, magma, MEMORYBUF, magma_t::output_resource_limits, and magma_t::system.

Referenced by process_start().

5.124.1.16 bool_t system_init_umask (void)

Set the process umask for new file/dir creation to O_RDWR.

Returns:

true if the umask was set successfully, or false on failure.

Definition at line 131 of file system.c.

References buflen, bufptr, and log_info.

Referenced by process_start().

5.124.1.17 uint64_t system_ulimit_cur (int_t resource)

Get the soft system limit for a specified resource.

Note:

This function will never return a value greater than UINT64_MAX.

See also:

getrlimit64()

Parameters:

resource the system rlimit resource identifier to be queried.

Returns:

-1 on failure, or the system soft limit of the specified resource identifier on success.

Definition at line 43 of file system.c.

References buflen, bufptr, log_info, and log_pedantic.

Referenced by st_realloc().

5.124.1.18 uint64_t system_ulimit_max (int_t resource)

Get the hard system limit for a specified resource.

Note:

This function will never return a value greater than UINT64_MAX.

See also:

getrlimit64()

Parameters:

resource the system rlimit resource identifier to be queried.

Returns:

-1 on failure, or the system hard limit of the specified resource identifier on success.

Definition at line 17 of file system.c.

References buflen, bufptr, log_info, and log_pedantic.

Referenced by config_validate_settings().

5.124.1.19 bool_t thread_start (void)

thread.c thread.c

Returns:

true on success or false on failure.

Definition at line 27 of file thread.c.

References signal_thread_start(), and sql_thread_start().

Referenced by dequeue(), net_accept(), and process_maint().

5.124.1.20 void thread_stop (void)

Prepare a thread to exit by destroying its MySQL and OpenSSL thread storage, and the thread specific mail cache.

Returns:

This function returns no value.

Definition at line 14 of file thread.c.

References mail_cache_thread_stop(), sql_thread_stop(), and ssl_thread_stop().

Referenced by dequeue(), net_accept(), and process_maint().

5.125 src/engine/context/sanity.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t sanity_check](#) (void)

Perform a series of system-wide sanity checks at process startup.

5.125.1 Function Documentation

5.125.1.1 bool_t sanity_check (void)

Perform a series of system-wide sanity checks at process startup.

Note:

This function ensures that the standard data types are their expected data sizes, and performs other checks, such as data primitive maximum values.

Returns:

true if the system environment passed all checks and should function correctly, or false if any checks failed.

Definition at line 15 of file sanity.c.

References [CONSTANT](#), [MAGMA_CORE_POOL_OBJECTS_LIMIT](#), [MAGMA_CORE_POOL_TIMEOUT_LIMIT](#), [NULLER](#), [placer_t](#), [queries](#), [record_t](#), [SELECT_DOMAINS](#), [st_alloc\(\)](#), [st_avail_get\(\)](#), [st_cmp_cs_eq\(\)](#), [st_data_get\(\)](#), and [st_free\(\)](#).

Referenced by [process_start\(\)](#).

5.126 src/engine/context/signal.c File Reference

```
#include "magma.h"
```

Functions

- void [signal_segfault](#) (int signal)
Handle signals that indicate the program was killed because of an invalid operation (including SIGSEGV).
- void [signal_shutdown](#) (int signal)
A function to handle receipt of shutdown signals and allow for graceful exits.
- void [signal_status](#) (int signal)
A generic worker thread signal handler entry point.
- void [signal_refresh](#) (int signal)
A function to handle receipt of SIGHUP and refresh all web server content.
- [bool_t](#) [signal_thread_start](#) (void)
Bind a SIGALRM handler for the calling thread.
- [bool_t](#) [signal_start](#) (void)
Setup signal masks and register signal handlers for handling shutdowns, program termination, and reloads.

Variables

- pthread_mutex_t [sig_hup_mutex](#) = PTHREAD_MUTEX_INITIALIZER

5.126.1 Function Documentation

5.126.1.1 void [signal_refresh](#) (int *signal*)

A function to handle receipt of SIGHUP and refresh all web server content.

Parameters:

signal the number of the delivered signal (should only be SIGHUP).

Returns:

This function returns no value.

Definition at line 107 of file signal.c.

References [http_content_refresh\(\)](#), [log_error](#), [log_info](#), [mutex_lock\(\)](#), [mutex_unlock\(\)](#), and [sig_hup_mutex](#).

Referenced by [signal_start\(\)](#).

5.126.1.2 void [signal_segfault](#) (int *signal*)

Handle signals that indicate the program was killed because of an invalid operation (including SIGSEGV).

Note:

The signals handled by this function are: SIGSEGV, SIGFPE, SIGBUS, SIGSYS, and SIGABRT. The handler will respond by logging a stack backtrace, and terminating the program with abort().

Parameters:

signal the number of the signal that was received.

Returns:

This function returns no value.

Definition at line 19 of file signal.c.

References log_options, M_LOG_CRITICAL, M_LOG_STACK_TRACE, mm_wipe(), and signal_name().

Referenced by signal_start().

5.126.1.3 void signal_shutdown (int *signal*)

A function to handle receipt of shutdown signals and allow for graceful exits.

Note:

This function handles the following signals: SIGINT, SIGQUIT, and SIGTERM. The shutdown procedure is as follows: 1. Set status to -1 and sleep for .1 second to allow for normal daemon termination. 2. Signal all worker threads to wake up blocked threads, and sleep one more second. 3. Loop through all possible file descriptors and if the descriptor matches a socket that is bound to a magma server, close it.

Parameters:

signal the number of the signal delivered to the process.

Returns:

This function returns no value.

Definition at line 52 of file signal.c.

References log_critical, MEMORYBUF, mm_wipe(), net_trigger(), queue_signal(), signal_name(), status_signal(), thread_join(), and thread_launch().

Referenced by signal_start().

5.126.1.4 bool_t signal_start (void)

Setup signal masks and register signal handlers for handling shutdowns, program termination, and reloads. [signal.c](#)

Note:

The following handlers are established: signal_shutdown: SIGINT, SIGQUIT, SIGTERM, SIGHUP signal_segfault: SIGSEGV, SIGFPE, SIGBUS, SIGSYS signal_refresh: SIGHUP [ignored]: SIGPIPE

Returns:

true if all signal handlers were successfully registered, or false on failure.

Definition at line 154 of file signal.c.

References log_info, mm_wipe(), signal_refresh(), signal_segfault(), and signal_shutdown().

Referenced by process_start().

5.126.1.5 void signal_status (int *signal*)

A generic worker thread signal handler entry point.

Note:

If the target thread is not in the process of shutting down, display the name of the caught signal.

Parameters:

signal the number of the signal caught by the signal handler.

Returns:

This function returns no value.

Definition at line 93 of file signal.c.

References log_info, signal_name(), and status.

Referenced by signal_thread_start().

5.126.1.6 bool_t signal_thread_start (void)

Bind a SIGALRM handler for the calling thread.

See also:

[signal_status\(\)](#)

Returns:

false on failure or true on success.

TODO: Disabling this signal handler to see if it fixes the shutdown issue.

Definition at line 128 of file signal.c.

References log_info, mm_wipe(), and signal_status().

Referenced by thread_start().

5.126.2 Variable Documentation

5.126.2.1 pthread_mutex_t sig_hup_mutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 10 of file signal.c.

Referenced by signal_refresh().

5.127 src/engine/context/system.c File Reference

```
#include "magma.h"
```

Functions

- [uint64_t system_ulimit_max \(int_t resource\)](#)
Get the hard system limit for a specified resource.
- [uint64_t system_ulimit_cur \(int_t resource\)](#)
Get the soft system limit for a specified resource.
- [bool_t system_change_root_directory \(void\)](#)
Perform a chroot() on the directory specified in the config option magma.system.root_directory, if it is set.
- [bool_t system_fork_daemon \(void\)](#)
Daemonize into the background, if the magma.system.daemonize config option is set.
- [bool_t system_init_umask \(void\)](#)
Set the process umask for new file/dir creation to O_RDWR.
- [bool_t system_init_core_dumps \(void\)](#)
Set the magma core dump size rlimit, if magma.system.enable_core_dumps was enabled.
- [bool_t system_init_impersonation \(void\)](#)
Set process privileges to run as a specified user if magma.system.impersonate_user is set.
- [bool_t system_init_resource_limits \(void\)](#)
Increase process resource limits, if magma.system.increase_resource_limits is set.

5.127.1 Function Documentation

5.127.1.1 bool_t system_change_root_directory (void)

Perform a chroot() on the directory specified in the config option magma.system.root_directory, if it is set. [system.c](#)

Returns:

true on success or false on failure.

Definition at line 66 of file system.c.

References [buflen](#), [bufptr](#), [log_info](#), [magma](#), [magma_t::root_directory](#), and [magma_t::system](#).

Referenced by [process_start\(\)](#).

5.127.1.2 bool_t system_fork_daemon (void)

Daemonize into the background, if the magma.system.daemonize config option is set.

Returns:

true inside the child process, or false inside the parent process or if an error occurs.

Definition at line 81 of file system.c.

References magma_t::daemonize, log_critical, log_info, magma, pid, status_process(), and magma_t::system.

Referenced by process_start().

5.127.1.3 bool_t system_init_core_dumps (void)

Set the magma core dump size rlimit, if magma.system.enable_core_dumps was enabled.

Returns:

true if core dumps were successfully enabled or false on failure.

Definition at line 149 of file system.c.

References buflen, bufptr, magma_t::core_dump_size_limit, magma_t::enable_core_dumps, log_info, magma, and magma_t::system.

Referenced by process_start().

5.127.1.4 bool_t system_init_impersonation (void)

Set process privileges to run as a specified user if magma.system.impersonate_user is set.

Note:

This function will set the user id and group id to the specified user, and chdir() to their home directory.

Returns:

true on success or if magma.system.impersonate_user is not set; false on failure to change privileges.

Definition at line 169 of file system.c.

References buflen, bufptr, magma_t::impersonate_user, log_critical, log_info, magma, mm_alloc(), mm_free(), mm_wipe(), status_process(), and magma_t::system.

Referenced by process_start().

5.127.1.5 bool_t system_init_resource_limits (void)

Increase process resource limits, if magma.system.increase_resource_limits is set.

Note:

Resource limits will be maximized for magma's virtual address space, data and stack segments, available file descriptors and sub-processes, and allowed file sizes. If output_resource_limits is enabled, the state of the process resource limits will be dumped to the log afterwards.

Returns:

This function always returns true (errors are logged).

Definition at line 244 of file system.c.

References magma_t::config, magma_t::increase_resource_limits, log_info, magma, MEMORYBUF, magma_t::output_resource_limits, and magma_t::system.

Referenced by process_start().

5.127.1.6 `bool_t system_init_umask (void)`

Set the process umask for new file/dir creation to O_RDWR.

Returns:

true if the umask was set successfully, or false on failure.

Definition at line 131 of file system.c.

References `buflen`, `bufptr`, and `log_info`.

Referenced by `process_start()`.

5.127.1.7 `uint64_t system_ulimit_cur (int_t resource)`

Get the soft system limit for a specified resource.

Note:

This function will never return a value greater than `UINT64_MAX`.

See also:

`getrlimit64()`

Parameters:

resource the system rlimit resource identifier to be queried.

Returns:

-1 on failure, or the system soft limit of the specified resource identifier on success.

Definition at line 43 of file system.c.

References `buflen`, `bufptr`, `log_info`, and `log_pedantic`.

Referenced by `st_realloc()`.

5.127.1.8 `uint64_t system_ulimit_max (int_t resource)`

Get the hard system limit for a specified resource.

Note:

This function will never return a value greater than `UINT64_MAX`.

See also:

`getrlimit64()`

Parameters:

resource the system rlimit resource identifier to be queried.

Returns:

-1 on failure, or the system hard limit of the specified resource identifier on success.

Definition at line 17 of file system.c.

References `buflen`, `bufptr`, `log_info`, and `log_pedantic`.

Referenced by `config_validate_settings()`.

5.128 src/engine/controller/controller.h File Reference

Functions

- void [dequeue](#) (void)
queue.c
- void [enqueue](#) (void *function, void *data)
Push a function on the job queue to be executed asynchronously.
- [bool_t](#) [queue_init](#) (void)
Create a queue of worker threads and set them into motion.
- void [queue_shutdown](#) (void)
Attempt to wake any sleeping workers, wait for all worker threads to shutdown and exit, and destroy the worker queue.
- void [queue_signal](#) (void)
Signal all worker threads with SIGALRM to force their return.
- void [requeue](#) (void *function, void *requeue, void *data)
Push a function on the job queue to be executed asynchronously.
- [bool_t](#) [protocol_init](#) (void)
protocol.c
- void [protocol_process](#) ([server_t](#) *server, int sockd)
Create a connection object for an accepted connection, and enqueue it to be handled.

5.128.1 Function Documentation

5.128.1.1 void dequeue (void)

[queue.c](#) [queue.c](#)

Note:

This is the thread pool entry point called from [queue_init\(\)](#).

Returns:

This function returns no value.

Definition at line 79 of file [queue.c](#).

References [queue_t::data](#), [queue_t::function](#), [log_error](#), [mm_free\(\)](#), [mutex_lock\(\)](#), [mutex_unlock\(\)](#), [queue_t::next](#), [queue](#), [queue_t::requeue\(\)](#), [stats_decrement_by_name\(\)](#), [stats_increment_by_name\(\)](#), [status](#), [thread_start\(\)](#), and [thread_stop\(\)](#).

Referenced by [queue_init\(\)](#).

5.128.1.2 void enqueue (void *function, void *data)

Push a function on the job queue to be executed asynchronously.

Note:

Warning: If this function fails to allocate a new [queue_t](#) object, the work unit is lost forever.

Parameters:

function a pointer to a function to be executed by the next available worker thread.

data a pointer to an arbitrary block of data to be passed to the specified function on execution.

Returns:

This function returns no value.

Definition at line 69 of file queue.c.

References [requeue\(\)](#).

Referenced by [con_reverse_enqueue\(\)](#), [dmtplib_process\(\)](#), [dmtplib_requeue\(\)](#), [http_process\(\)](#), [http_requeue\(\)](#), [imap_process\(\)](#), [imap_requeue\(\)](#), [molten_init\(\)](#), [molten_invalid\(\)](#), [molten_parse\(\)](#), [molten_stats\(\)](#), [pop_process\(\)](#), [pop_requeue\(\)](#), [protocol_enqueue\(\)](#), [protocol_process\(\)](#), [smtp_process\(\)](#), and [smtp_requeue\(\)](#).

5.128.1.3 bool_t protocol_init (void)

[protocol.c](#) [protocol.c](#)

Returns:

This function always returns true.

Definition at line 86 of file protocol.c.

References [dmtplib_sort\(\)](#), [imap_sort\(\)](#), [molten_sort\(\)](#), [pop_sort\(\)](#), [portal_endpoint_sort\(\)](#), and [smtp_sort\(\)](#).

Referenced by [process_start\(\)](#).

5.128.1.4 void protocol_process (server_t * server, int sockd)

Create a connection object for an accepted connection, and enqueue it to be handled.

See also:

[protocol_secure\(\)](#), [protocol_enqueue\(\)](#)

Note:

Depending on whether the server specifies a secure transport layer, [protocol_secure\(\)](#) or [protocol_enqueue\(\)](#) will be dispatched.

Parameters:

server a pointer to the server object of the server handling the connection.

sockd the socket descriptor of the newly accepted connection.

Definition at line 201 of file protocol.c.

References [con_init\(\)](#), [connection_t](#), [server_t::context](#), [enqueue\(\)](#), [log_pedantic\(\)](#), [net_set_timeout\(\)](#), [server_t::network](#), [protocol_enqueue\(\)](#), [protocol_secure\(\)](#), [server_t::timeout](#), [server_t::tls](#), [TLS_PORT](#), and [server_t::type](#).

Referenced by [net_accept\(\)](#).

5.128.1.5 bool_t queue_init (void)

Create a queue of worker threads and set them into motion.

Note:

Up to magma.system.worker_threads number of threads will be created.

Returns:

false on failure or true on success.

Definition at line 154 of file queue.c.

References dequeue(), log_error, magma, mm_alloc(), mutex_init(), queue, queue_shutdown(), stats_increment_by_name(), magma_t::system, thread_launch(), and magma_t::worker_threads.

Referenced by process_start().

5.128.1.6 void queue_shutdown (void)

Attempt to wake any sleeping workers, wait for all worker threads to shutdown and exit, and destroy the worker queue.

Returns:

This function returns no value.

Definition at line 188 of file queue.c.

References magma, mm_cleanup, mutex_destroy(), queue, stats_decrement_by_name(), magma_t::system, thread_join(), and magma_t::worker_threads.

Referenced by process_stop(), and queue_init().

5.128.1.7 void queue_signal (void)

Signal all worker threads with SIGALRM to force their return.

Returns:

This function returns no value.

Definition at line 133 of file queue.c.

References log_info, magma, queue, status, magma_t::system, thread_signal(), and magma_t::worker_threads.

Referenced by net_trigger(), and signal_shutdown().

5.128.1.8 void requeue (void *function, void *requeue, void *data)

Push a function on the job queue to be executed asynchronously.

Note:

Warning: If this function fails to allocate a new [queue_t](#) object, the work unit is lost forever.

Parameters:

function a pointer to a function to be executed by the next available worker thread.

requeue an optional pointer to a requeue function to be called after function is executed.

data a pointer to an arbitrary block of data to be passed to function and/or requeue upon execution.

Returns:

This function returns no value.

Definition at line 33 of file queue.c.

References `queue_t::data`, `queue_t::function`, `log_critical`, `mm_alloc()`, `mutex_lock()`, `mutex_unlock()`, `queue_t::next`, `queue`, and `queue_t::requeue()`.

Referenced by `dmtplib_process()`, `enqueue()`, `http_process()`, `http_requeue()`, `imap_process()`, `pop_process()`, `sess_release()`, `smtp_data()`, and `smtp_process()`.

5.129 src/engine/controller/protocol.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * protocol_type](#) ([connection_t *con](#))
- [bool_t protocol_init](#) (void)
Initialize all protocol modules, and prime their command arrays for binary searching.
- void [protocol_enqueue](#) ([connection_t *con](#))
Enqueue a protocol-specific handler to service a specified connection, and update any statistics accordingly.
- void [protocol_secure](#) ([connection_t *con](#))
Establish a secure channel for an inbound TLS connection before passing it off to the general server request handler.
- void [protocol_process](#) ([server_t *server](#), int sockd)
Create a connection object for an accepted connection, and enqueue it to be handled.

5.129.1 Function Documentation

5.129.1.1 void protocol_enqueue (connection_t * con)

Enqueue a protocol-specific handler to service a specified connection, and update any statistics accordingly.

Note:

If an invalid protocol is specified, the connection will be destroyed gracefully.

Parameters:

con a pointer to the connection object to be serviced.

Returns:

This function returns no value.

Definition at line 102 of file protocol.c.

References [con_destroy\(\)](#), [con_secure\(\)](#), [DMTP](#), [dmtplib_init\(\)](#), [enqueue\(\)](#), [HTTP](#), [http_init\(\)](#), [IMAP](#), [imap_init\(\)](#), [log_check](#), [log_pedantic](#), [MOLTEN](#), [molten_init\(\)](#), [POP](#), [pop_init\(\)](#), [SMTP](#), [smtp_init\(\)](#), [stats_increment_by_name\(\)](#), [SUBMISSION](#), and [submission_init\(\)](#).

Referenced by [protocol_process\(\)](#), and [protocol_secure\(\)](#).

5.129.1.2 bool_t protocol_init (void)

Initialize all protocol modules, and prime their command arrays for binary searching. [protocol.c](#)

Returns:

This function always returns true.

Definition at line 86 of file protocol.c.

References [dmtplib_sort\(\)](#), [imap_sort\(\)](#), [molten_sort\(\)](#), [pop_sort\(\)](#), [portal_endpoint_sort\(\)](#), and [smtp_sort\(\)](#).

Referenced by [process_start\(\)](#).

5.129.1.3 void protocol_process (server_t * server, int sockd)

Create a connection object for an accepted connection, and enqueue it to be handled.

See also:

[protocol_secure\(\)](#), [protocol_enqueue\(\)](#)

Note:

Depending on whether the server specifies a secure transport layer, [protocol_secure\(\)](#) or [protocol_enqueue\(\)](#) will be dispatched.

Parameters:

server a pointer to the server object of the server handling the connection.

sockd the socket descriptor of the newly accepted connection.

Definition at line 201 of file protocol.c.

References [con_init\(\)](#), [connection_t](#), [server_t::context](#), [enqueue\(\)](#), [log_pedantic](#), [net_set_timeout\(\)](#), [server_t::network](#), [protocol_enqueue\(\)](#), [protocol_secure\(\)](#), [server_t::timeout](#), [server_t::tls](#), [TLS_PORT](#), and [server_t::type](#).

Referenced by [net_accept\(\)](#).

5.129.1.4 void protocol_secure (connection_t * con)

Establish a secure channel for an inbound TLS connection before passing it off to the general server request handler.

See also:

[protocol_enqueue\(\)](#)

Note:

This function destroys the client connection completely and returns silently upon any TLS-related failure.

Parameters:

con the The connection object associated with the inbound TLS connection.

Returns:

This function returns no value.

Definition at line 165 of file protocol.c.

References [con_addr_presentation\(\)](#), [log_check](#), [log_pedantic](#), [M_SSL_BIO_NOCLOSE](#), [MANAGEDBUF](#), [mm_free\(\)](#), [mutex_destroy\(\)](#), [protocol_enqueue\(\)](#), [protocol_type\(\)](#), [st_char_get\(\)](#), [st_free\(\)](#), [st_length_int\(\)](#), [status](#), [tls_free\(\)](#), and [tls_server_alloc\(\)](#).

Referenced by [protocol_process\(\)](#).

5.129.1.5 stringer_t* protocol_type (connection_t * con)

Definition at line 11 of file protocol.c.

References [CONSTANT](#), [DMTP](#), [HTTP](#), [IMAP](#), [log_check](#), [MOLTEN](#), [POP](#), [SMTP](#), [SUBMISSION](#), and [TLS_PORT](#).

Referenced by [protocol_secure\(\)](#).

5.130 src/engine/controller/queue.c File Reference

```
#include "magma.h"
```

Data Structures

- struct [queue_t](#)

Functions

- void [requeue](#) (void *function, void *requeue, void *data)
Push a function on the job queue to be executed asynchronously.
- void [enqueue](#) (void *function, void *data)
Push a function on the job queue to be executed asynchronously.
- void [dequeue](#) (void)
Wait for work to appear on the queue and then perform the work; if the job is to be requeue'd then requeue it.
- void [queue_signal](#) (void)
Signal all worker threads with SIGALRM to force their return.
- [bool_t queue_init](#) (void)
Create a queue of worker threads and set them into motion.
- void [queue_shutdown](#) (void)
Attempt to wake any sleeping workers, wait for all worker threads to shutdown and exit, and destroy the worker queue.

Variables

- struct {
 sem_t [sema](#)
 pthread_t * [workers](#)
 pthread_mutex_t [lock](#)
 [queue_t](#) * [items](#)
} [queue](#)

5.130.1 Function Documentation

5.130.1.1 void dequeue (void)

Wait for work to appear on the queue and then perform the work; if the job is to be requeue'd then requeue it. [queue.c](#)

Note:

This is the thread pool entry point called from [queue_init\(\)](#).

Returns:

This function returns no value.

Definition at line 79 of file queue.c.

References queue_t::data, queue_t::function, log_error, mm_free(), mutex_lock(), mutex_unlock(), queue_t::next, queue, queue_t::requeue(), stats_decrement_by_name(), stats_increment_by_name(), status, thread_start(), and thread_stop().

Referenced by queue_init().

5.130.1.2 void enqueue (void * *function*, void * *data*)

Push a function on the job queue to be executed asynchronously.

Note:

Warning: If this function fails to allocate a new [queue_t](#) object, the work unit is lost forever.

Parameters:

function a pointer to a function to be executed by the next available worker thread.

data a pointer to an arbitrary block of data to be passed to the specified function on execution.

Returns:

This function returns no value.

Definition at line 69 of file queue.c.

References requeue().

Referenced by con_reverse_enqueue(), dmtip_process(), dmtip_requeue(), http_process(), http_requeue(), imap_process(), imap_requeue(), molten_init(), molten_invalid(), molten_parse(), molten_stats(), pop_process(), pop_requeue(), protocol_enqueue(), protocol_process(), smtp_process(), and smtp_requeue().

5.130.1.3 bool_t queue_init (void)

Create a queue of worker threads and set them into motion.

Note:

Up to magma.system.worker_threads number of threads will be created.

Returns:

false on failure or true on success.

Definition at line 154 of file queue.c.

References dequeue(), log_error, magma, mm_alloc(), mutex_init(), queue, queue_shutdown(), stats_increment_by_name(), magma_t::system, thread_launch(), and magma_t::worker_threads.

Referenced by process_start().

5.130.1.4 void queue_shutdown (void)

Attempt to wake any sleeping workers, wait for all worker threads to shutdown and exit, and destroy the worker queue.

Returns:

This function returns no value.

Definition at line 188 of file queue.c.

References magma, mm_cleanup, mutex_destroy(), queue, stats_decrement_by_name(), magma_t::system, thread_join(), and magma_t::worker_threads.

Referenced by process_stop(), and queue_init().

5.130.1.5 void queue_signal (void)

Signal all worker threads with SIGALRM to force their return.

Returns:

This function returns no value.

Definition at line 133 of file queue.c.

References log_info, magma, queue, status, magma_t::system, thread_signal(), and magma_t::worker_threads.

Referenced by net_trigger(), and signal_shutdown().

5.130.1.6 void requeue (void *function, void *requeue, void *data)

Push a function on the job queue to be executed asynchronously.

Note:

Warning: If this function fails to allocate a new [queue_t](#) object, the work unit is lost forever.

Parameters:

function a pointer to a function to be executed by the next available worker thread.

requeue an optional pointer to a requeue function to be called after function is executed.

data a pointer to an arbitrary block of data to be passed to function and/or requeue upon execution.

Returns:

This function returns no value.

Definition at line 33 of file queue.c.

References queue_t::data, queue_t::function, log_critical, mm_alloc(), mutex_lock(), mutex_unlock(), queue_t::next, queue, and queue_t::requeue().

Referenced by dmtip_process(), enqueue(), http_process(), http_requeue(), imap_process(), pop_process(), sess_release(), smtp_data(), and smtp_process().

5.130.2 Variable Documentation

5.130.2.1 queue_t* items

Definition at line 19 of file queue.c.

5.130.2.2 pthread_mutex_t lock

Definition at line 18 of file queue.c.

5.130.2.3 struct { ... } queue

Referenced by dequeue(), queue_init(), queue_shutdown(), queue_signal(), and requeue().

5.130.2.4 sem_t sema

Definition at line 16 of file queue.c.

5.130.2.5 pthread_t* workers

Definition at line 17 of file queue.c.

5.131 src/engine/engine.h File Reference

```
#include "log/log.h"
#include "context/context.h"
#include "config/config.h"
#include "status/status.h"
#include "controller/controller.h"
```

5.132 src/engine/log/log.c File Reference

```
#include "magma.h"
```

Functions

- void [log_disable](#) (void)
Disable logging.
- void [log_enable](#) (void)
Enable logging.
- [int_t log_backtrace](#) (FILE *output)
Log the current stack to stdout.
- void [log_internal](#) (const char *file, const char *function, const int line, [M_LOG_OPTIONS](#) options, const char *format,...)
Logs the message described by format, and provided as a variadic argument list.
- void [log_rotate](#) (void)
- [bool_t log_start](#) (void)

Variables

- [uint64_t log_date](#)
- [bool_t log_enabled](#) = true
- FILE * [log_descriptor](#) = NULL
- [pthread_mutex_t log_mutex](#) = PTHREAD_MUTEX_INITIALIZER

5.132.1 Function Documentation

5.132.1.1 [int_t log_backtrace](#) (FILE * *output*)

Log the current stack to stdout.

Note:

This function was created because `backtrace_symbols()` can fail if there is heap corruption.

Returns:

-1 if the backtrace failed, or 0 on success.

Definition at line 42 of file `log.c`.

Referenced by `log_internal()`.

5.132.1.2 [void log_disable](#) (void)

Disable logging.

Returns:

This function returns no value.

Definition at line 19 of file `log.c`.

References `log_enabled`, `log_mutex`, `mutex_lock()`, and `mutex_unlock()`.

5.132.1.3 void log_enable (void)

Enable logging.

Returns:

This function returns no value.

Definition at line 30 of file log.c.

References log_enabled, log_mutex, mutex_lock(), and mutex_unlock().

5.132.1.4 void log_internal (const char **file*, const char **function*, const int *line*, M_LOG_OPTIONS *options*, const char **format*, ...)

Logs the message described by format, and provided as a variadic argument list.

Parameters:

file The log macros set this to the caller's filename.

function The log macros set this to the caller's function.

line The log macros set this to the line number where the log function was called.

options Global configuration options can be overridden on a per call basis using the options variable.

format The printf style format for the log message.

va_list A variadic list of data items to be used by the format string.

Returns:

This function returns no value.

Definition at line 112 of file log.c.

References magma_t::file, magma_t::function, magma_t::line, magma_t::log, log_backtrace(), log_descriptor, log_enabled, log_mutex, M_LOG_CONSOLE, M_LOG_FILE, M_LOG_FILE_DISABLE, M_LOG_FUNCTION, M_LOG_FUNCTION_DISABLE, M_LOG_LINE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE, M_LOG_STACK_TRACE_DISABLE, M_LOG_TIME, M_LOG_TIME_DISABLE, magma, mutex_lock(), mutex_unlock(), magma_t::output, magma_t::stack, and magma_t::time.

Referenced by xml_error().

5.132.1.5 void log_rotate (void)

Definition at line 202 of file log.c.

References magma_t::daemonize, magma_t::file, log_critical, log_date, log_descriptor, log_mutex, magma, MEMORYBUF, ns_length_get(), magma_t::output, magma_t::path, magma_t::system, and time_datestamp().

Referenced by process_maint().

5.132.1.6 bool_t log_start (void)

Definition at line 255 of file log.c.

References magma_t::daemonize, magma_t::file, folder_exists(), log_critical, log_date, log_descriptor, magma, MEMORYBUF, ns_length_get(), NULLER, magma_t::output, magma_t::path, magma_t::system, and time_datestamp().

Referenced by process_start().

5.132.2 Variable Documentation

5.132.2.1 `uint64_t log_date`

Definition at line 10 of file `log.c`.

Referenced by `log_rotate()`, and `log_start()`.

5.132.2.2 `FILE* log_descriptor = NULL`

Definition at line 12 of file `log.c`.

Referenced by `log_internal()`, `log_rotate()`, `log_start()`, and `main()`.

5.132.2.3 `bool_t log_enabled = true`

Definition at line 11 of file `log.c`.

Referenced by `log_disable()`, `log_enable()`, and `log_internal()`.

5.132.2.4 `pthread_mutex_t log_mutex = PTHREAD_MUTEX_INITIALIZER`

Definition at line 13 of file `log.c`.

Referenced by `log_disable()`, `log_enable()`, `log_internal()`, and `log_rotate()`.

5.133 src/engine/log/log.h File Reference

Defines

- #define [MAGMA_LOG_LEVELS](#) (M_LOG_PEDANTIC | M_LOG_INFO | M_LOG_WARN | M_LOG_ERROR | M_LOG_CRITICAL)
- #define [log_pedantic](#)(...) do { } while (0)
- #define [log_check](#)(expr) do { } while (0)
- #define [log_info](#)(...) log_internal (__FILE__, __FUNCTION__, __LINE__, M_LOG_INFO, __VA_ARGS__)
- #define [log_warn](#)(...) log_internal (__FILE__, __FUNCTION__, __LINE__, M_LOG_WARN, __VA_ARGS__)
- #define [log_error](#)(...) log_internal (__FILE__, __FUNCTION__, __LINE__, M_LOG_ERROR, __VA_ARGS__)
- #define [log_critical](#)(...) log_internal (__FILE__, __FUNCTION__, __LINE__, M_LOG_CRITICAL, __VA_ARGS__)
- #define [log_options](#)(options,...) log_internal (__FILE__, __FUNCTION__, __LINE__, options, __VA_ARGS__)

Enumerations

- enum [M_LOG_OPTIONS](#) {
[M_LOG_PEDANTIC](#) = 1, [M_LOG_INFO](#) = 2, [M_LOG_WARN](#) = 4, [M_LOG_ERROR](#) = 8,
[M_LOG_CRITICAL](#) = 16, [M_LOG_TIME](#) = 32, [M_LOG_FILE](#) = 64, [M_LOG_LINE](#) = 128,
[M_LOG_FUNCTION](#) = 256, [M_LOG_STACK_TRACE](#) = 512, [M_LOG_CONSOLE](#) = 1024, [M_LOG_PEDANTIC_DISABLE](#) = 2048,
[M_LOG_INFO_DISABLE](#) = 4096, [M_LOG_WARN_DISABLE](#) = 8192, [M_LOG_ERROR_DISABLE](#) = 16384, [M_LOG_CRITICAL_DISABLE](#) = 32768,
[M_LOG_LINE_FEED_DISABLE](#) = 65536, [M_LOG_TIME_DISABLE](#) = 131072, [M_LOG_FILE_DISABLE](#) = 262144, [M_LOG_LINE_DISABLE](#) = 524288,
[M_LOG_FUNCTION_DISABLE](#) = 1048576, [M_LOG_STACK_TRACE_DISABLE](#) = 2097152 }

Functions

- void [log_internal](#) (const char *file, const char *function, const int line, [M_LOG_OPTIONS](#) options, const char *format,...) [__attribute__\(\(format\(printf](#)
- void [log_disable](#) (void)
Disable logging.
- void [log_enable](#) (void)
Enable logging.
- void [log_rotate](#) (void)
- [bool_t](#) [log_start](#) (void)
- [int_t](#) [log_backtrace](#) (FILE *output)
Log the current stack to stdout.

5.133.1 Define Documentation

5.133.1.1 #define [log_check](#)(expr) do { } while (0)

Definition at line 77 of file log.h.

5.133.1.2 `#define log_critical(...) log_internal (__FILE__, __FUNCTION__, __LINE__, M_LOG_CRITICAL, __VA_ARGS__)`

Definition at line 94 of file log.h.

5.133.1.3 `#define log_error(...) log_internal (__FILE__, __FUNCTION__, __LINE__, M_LOG_ERROR, __VA_ARGS__)`

Definition at line 91 of file log.h.

5.133.1.4 `#define log_info(...) log_internal (__FILE__, __FUNCTION__, __LINE__, M_LOG_INFO, __VA_ARGS__)`

Definition at line 83 of file log.h.

5.133.1.5 `#define log_options(options, ...) log_internal (__FILE__, __FUNCTION__, __LINE__, options, __VA_ARGS__)`

Definition at line 97 of file log.h.

5.133.1.6 `#define log_pedantic(...) do {} while (0)`

Definition at line 76 of file log.h.

5.133.1.7 `#define log_warn(...) log_internal (__FILE__, __FUNCTION__, __LINE__, M_LOG_WARN, __VA_ARGS__)`

Definition at line 88 of file log.h.

Referenced by config_load_cmdline_settings(), config_load_database_settings(), and config_validate_settings().

5.133.1.8 `#define MAGMA_LOG_LEVELS (M_LOG_PEDANTIC | M_LOG_INFO | M_LOG_WARN | M_LOG_ERROR | M_LOG_CRITICAL)`

Definition at line 49 of file log.h.

5.133.2 Enumeration Type Documentation

5.133.2.1 enum M_LOG_OPTIONS

Enumerator:

M_LOG_PEDANTIC

M_LOG_INFO

M_LOG_WARN

M_LOG_ERROR

M_LOG_CRITICAL

M_LOG_TIME

M_LOG_FILE

M_LOG_LINE

M_LOG_FUNCTION

M_LOG_STACK_TRACE

M_LOG_CONSOLE

M_LOG_PEDANTIC_DISABLE

M_LOG_INFO_DISABLE
M_LOG_WARN_DISABLE
M_LOG_ERROR_DISABLE
M_LOG_CRITICAL_DISABLE
M_LOG_LINE_FEED_DISABLE
M_LOG_TIME_DISABLE
M_LOG_FILE_DISABLE
M_LOG_LINE_DISABLE
M_LOG_FUNCTION_DISABLE
M_LOG_STACK_TRACE_DISABLE

Definition at line 14 of file log.h.

5.133.3 Function Documentation

5.133.3.1 `int_t log_backtrace (FILE * output)`

Log the current stack to stdout.

Note:

This function was created because `backtrace_symbols()` can fail if there is heap corruption.

Returns:

-1 if the backtrace failed, or 0 on success.

Definition at line 42 of file log.c.

Referenced by `log_internal()`.

5.133.3.2 `void void log_disable (void)`

Disable logging.

Returns:

This function returns no value.

Definition at line 19 of file log.c.

References `log_enabled`, `log_mutex`, `mutex_lock()`, and `mutex_unlock()`.

5.133.3.3 `void log_enable (void)`

Enable logging.

Returns:

This function returns no value.

Definition at line 30 of file log.c.

References `log_enabled`, `log_mutex`, `mutex_lock()`, and `mutex_unlock()`.

5.133.3.4 `void log_internal (const char *file, const char *function, const int line, M_LOG_OPTIONS options, const char *format, ...)`

5.133.3.5 `void log_rotate (void)`

Definition at line 202 of file log.c.

References magma_t::daemonize, magma_t::file, log_critical, log_date, log_descriptor, log_mutex, magma, MEMORYBUF, ns_length_get(), magma_t::output, magma_t::path, magma_t::system, and time_datestamp().

Referenced by process_maint().

5.133.3.6 `bool_t log_start (void)`

Definition at line 255 of file log.c.

References magma_t::daemonize, magma_t::file, folder_exists(), log_critical, log_date, log_descriptor, magma, MEMORYBUF, ns_length_get(), NULLER, magma_t::output, magma_t::path, magma_t::system, and time_datestamp().

Referenced by process_start().

5.134 src/engine/status/build.c File Reference

```
#include "magma.h"
```

Functions

- `const char * build_version (void)`
Get the magma version string.
- `const char * build_commit (void)`
Get the magma commit identity.
- `const char * build_stamp (void)`
Get the magma build stamp.
- `uint64_t build_version_major (void)`
Get the magma major version number.
- `uint64_t build_version_minor (void)`
Get the magma minor version number.
- `uint64_t build_version_patch (void)`
Get the magma patch level.

5.134.1 Function Documentation

5.134.1.1 `const char* build_commit (void)`

Get the magma commit identity. [build.c](#)

Returns:

a pointer to a null-terminated string containing the last eight characters of current commit identity.

Definition at line 34 of file `build.c`.

Referenced by `args_parse()`, and `lib_load()`.

5.134.1.2 `const char* build_stamp (void)`

Get the magma build stamp.

Returns:

a pointer to a null-terminated string containing the magma build information string.

Definition at line 42 of file `build.c`.

Referenced by `args_parse()`, `http_response_options()`, `imap_id()`, and `lib_load()`.

5.134.1.3 `const char* build_version (void)`

Get the magma version string.

Returns:

a pointer to a null-terminated string containing the magma version string.

Definition at line 26 of file build.c.

Referenced by `args_parse()`, `http_response_options()`, `imap_id()`, `imap_init()`, `lib_load()`, and `pop_capa()`.

5.134.1.4 `uint64_t build_version_major (void)`

Get the magma major version number.

Returns:

an unsigned integer with the major version number.

Definition at line 50 of file build.c.

References `ns_length_get()`, `pl_null()`, `PLACER`, `placer_t`, `tok_get_st()`, and `uint64_conv_pl()`.

5.134.1.5 `uint64_t build_version_minor (void)`

Get the magma minor version number.

Returns:

an unsigned integer with the minor version number.

Definition at line 66 of file build.c.

References `ns_length_get()`, `pl_null()`, `PLACER`, `placer_t`, `tok_get_st()`, and `uint64_conv_pl()`.

5.134.1.6 `uint64_t build_version_patch (void)`

Get the magma patch level.

Returns:

an unsigned integer with the patch level.

Definition at line 82 of file build.c.

References `ns_length_get()`, `pl_null()`, `PLACER`, `placer_t`, `tok_get_st()`, and `uint64_conv_pl()`.

5.135 src/engine/status/performance.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t perf_rdtsc` (void)
performance.c

5.135.1 Function Documentation

5.135.1.1 `uint64_t perf_rdtsc` (void)

[performance.c](#) Returns the CPU cycle counter. By calculating variations in the time stamp, we can discover the exact number of elapsed CPU ticks, making it possible to precisely time code execution. Note that in a multithreaded environment its possible that not all of the elapsed ticks were consumed by the code being timed.

Returns:

The number of CPU cycles that have elapsed since boot.

Definition at line 180 of file `performance.c`.

5.136 src/engine/status/statistics.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t stats_sum_errors` (void)
Get the total sum of all error statistics maintained by magma (traditional and derived stats).
- `uint64_t stats_derived_count` (void)
Get the number of derived statistics that are tracked.
- `char * stats_derived_name` (uint64_t position)
Get the name of a derived statistic by index.
- `uint64_t stats_derived_value` (uint64_t position)
Get the value of a derived statistic by index.
- `uint64_t stats_get_name_pos` (char *name)
Get the index of a statistic by name.
- `char * stats_get_name` (uint64_t position)
Get the name of a statistic by its index.
- `void stats_set_by_name` (char *name, uint64_t value)
Provided a statistic by name, set its value.
- `void stats_set_by_num` (uint64_t position, uint64_t value)
Provided a statistic by index, set its value.
- `uint64_t stats_get_value_by_name` (char *name)
Provided a statistic by name, get its value.
- `uint64_t stats_get_value_by_num` (uint64_t position)
Provided a statistic by index, get its value.
- `void stats_adjust_by_name` (char *name, int32_t value)
Provided a statistic by name, increment its value by a specified amount.
- `void stats_adjust_by_num` (uint64_t position, int32_t value)
Provided a statistic by index, increment its value by a specified amount.
- `void stats_increment_by_name` (char *name)
Provided a statistic by name, increment its value by 1.
- `void stats_increment_by_num` (uint64_t position)
Provided a statistic by index, increment its value by 1.
- `void stats_decrement_by_name` (char *name)
Provided a statistic by name, decrement its value by 1.
- `void stats_decrement_by_num` (uint64_t position)

Provided a statistic by index, decrement its value by 1.

- `uint64_t stats_get_count (void)`
Get the number of statistics being tracked.
- `bool_t stats_init (void)`
Initialize and reset all statistics counters.
- `void stats_shutdown (void)`
Destroy all statistics locks.

Variables

- struct {
 size_t `count`
 pthread_mutex_t `locks` [128][2]
 uint64_t `values` [128]
 char * `names` [128]
} `stats`
- char * `derived` []

5.136.1 Function Documentation

5.136.1.1 void stats_adjust_by_name (char * *name*, int32_t *value*)

Provided a statistic by name, increment its value by a specified amount. statistics.c

Parameters:

- name*** a null-terminated string containing the name of the statistic to be set.
value the value by which to increment the specified statistic.

Returns:

This function returns no value.

Definition at line 360 of file statistics.c.

References `mutex_lock()`, `mutex_unlock()`, `stats`, and `stats_get_name_pos()`.

Referenced by `dkim_signature_create()`, `dkim_signature_verify()`, `obj_cache_prune()`, `pattern_check()`, and `spf_check()`.

5.136.1.2 void stats_adjust_by_num (uint64_t *position*, int32_t *value*)

Provided a statistic by index, increment its value by a specified amount.

Parameters:

- position*** the zero-based index of the statistic to be set.
value the value by which to increment the specified statistic.

Returns:

This function returns no value.

Definition at line 381 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

5.136.1.3 void stats_decrement_by_name (char * *name*)

Provided a statistic by name, decrement its value by 1.

Parameters:

name a null-terminated string containing the name of the statistic to be decremented.

Returns:

This function returns no value.

Definition at line 429 of file statistics.c.

References mutex_lock(), mutex_unlock(), stats, and stats_get_name_pos().

Referenced by con_destroy(), dequeue(), and queue_shutdown().

5.136.1.4 void stats_decrement_by_num (uint64_t *position*)

Provided a statistic by index, decrement its value by 1.

Parameters:

position the zero-based index of the statistic to be decremented.

Returns:

This function returns no value.

Definition at line 449 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

5.136.1.5 uint64_t stats_derived_count (void)

Get the number of derived statistics that are tracked.

Returns:

the number of derived statistics being maintained by magma.

Definition at line 183 of file statistics.c.

References derived.

Referenced by molten_stats(), stats_derived_name(), and stats_sum_errors().

5.136.1.6 char* stats_derived_name (uint64_t *position*)

Get the name of a derived statistic by index.

Parameters:

position the zero-based index of the derived statistic to be queried.

Returns:

NULL on failure, or a pointer to a null-terminated string containing the name of the derived statistic on success.

Definition at line 193 of file statistics.c.

References `derived`, and `stats_derived_count()`.

Referenced by `molten_stats()`, and `stats_sum_errors()`.

5.136.1.7 uint64_t stats_derived_value (uint64_t position)

Get the value of a derived statistic by index.

See also:

[mm_sec_stats\(\)](#)

Parameters:

position the zero-based index of the derived statistic to be queried.

Returns:

0 on failure, or the value of the specified derived statistic on success.

Definition at line 208 of file statistics.c.

References `bytes`, `items`, `log_pedantic`, `mm_sec_stats()`, `spool_error_stats()`, and `stats_sum_errors()`.

Referenced by `molten_stats()`, and `stats_sum_errors()`.

5.136.1.8 uint64_t stats_get_count (void)

Get the number of statistics being tracked.

Returns:

the total number of statistics counters maintained by magma.

Definition at line 462 of file statistics.c.

References `stats`.

Referenced by `molten_stats()`, and `stats_sum_errors()`.

5.136.1.9 char* stats_get_name (uint64_t position)

Get the name of a statistic by its index.

Parameters:

position the zero-based index of the statistic to be queried.

Returns:

0 on failure, or the name of the requested statistic on success.

Definition at line 272 of file statistics.c.

References `mutex_lock()`, `mutex_unlock()`, and `stats`.

Referenced by `molten_stats()`, and `stats_sum_errors()`.

5.136.1.10 `uint64_t stats_get_name_pos (char * name)`

Get the index of a statistic by name.

Parameters:

name the name of the statistic to be queried.

Returns:

0 on failure, or the zero-based index of the requested statistic on success.

Definition at line 251 of file statistics.c.

References `log_info`, `mutex_lock()`, `mutex_unlock()`, `NULLER`, `st_cmp_cs_eq()`, and `stats`.

Referenced by `stats_adjust_by_name()`, `stats_decrement_by_name()`, `stats_get_value_by_name()`, `stats_increment_by_name()`, and `stats_set_by_name()`.

5.136.1.11 `uint64_t stats_get_value_by_name (char * name)`

Provided a statistic by name, get its value.

Parameters:

name a null-terminated string containing the name of the statistic to be queried.

Returns:

the specified statistic's value, as an unsigned 64 bit integer.

Definition at line 323 of file statistics.c.

References `mutex_lock()`, `mutex_unlock()`, `stats`, and `stats_get_name_pos()`.

5.136.1.12 `uint64_t stats_get_value_by_num (uint64_t position)`

Provided a statistic by index, get its value.

Parameters:

position the zero-based index of the statistic to be queried.

Returns:

the specified statistic's value, as an unsigned 64 bit integer.

Definition at line 343 of file statistics.c.

References `mutex_lock()`, `mutex_unlock()`, and `stats`.

Referenced by `molten_stats()`, and `stats_sum_errors()`.

5.136.1.13 `void stats_increment_by_name (char * name)`

Provided a statistic by name, increment its value by 1.

Parameters:

name a null-terminated string containing the name of the statistic to be incremented.

Returns:

This function returns no value.

Definition at line 395 of file statistics.c.

References mutex_lock(), mutex_unlock(), stats, and stats_get_name_pos().

Referenced by dequeue(), imap_starttls(), pop_starttls(), protocol_enqueue(), queue_init(), register_abuse_check_blocklist(), smtp_starttls(), virus_check(), virus_engine_refresh(), and virus_start().

5.136.1.14 void stats_increment_by_num (uint64_t *position*)

Provided a statistic by index, increment its value by 1.

Parameters:

position the zero-based index of the statistic to be incremented.

Returns:

This function returns no value.

Definition at line 415 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

5.136.1.15 bool_t stats_init (void)

Initialize and reset all statistics counters.

Returns:

false on failure or true on success.

Definition at line 471 of file statistics.c.

References log_critical, mm_wipe(), mutex_init(), and stats.

Referenced by process_start().

5.136.1.16 void stats_set_by_name (char * *name*, uint64_t *value*)

Provided a statistic by name, set its value.

Parameters:

name a null-terminated string containing the name of the statistic to be set.

value the new value of the specified statistic.

Returns:

This function returns no value.

Definition at line 289 of file statistics.c.

References mutex_lock(), mutex_unlock(), stats, and stats_get_name_pos().

Referenced by obj_cache_prune(), virus_engine_refresh(), virus_start(), and virus_stop().

5.136.1.17 void stats_set_by_num (uint64_t position, uint64_t value)

Provided a statistic by index, set its value.

Parameters:

position the zero-based index of the statistic to be set.

value the new value of the specified statistic.

Returns:

This function returns no value.

Definition at line 309 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

5.136.1.18 void stats_shutdown (void)

Destroy all statistics locks.

Returns:

This function returns no value.

Definition at line 494 of file statistics.c.

References mutex_destroy(), and stats.

Referenced by process_stop().

5.136.1.19 uint64_t stats_sum_errors (void)

Get the total sum of all error statistics maintained by magma (traditional and derived stats).

Note:

Error statistics are all statistics that have a name that begins with "errors." or ends with ".errors".

Returns:

The total sum of all error statistics counters accumulated by magma.

Definition at line 137 of file statistics.c.

References CONSTANT, NULLER, st_cmp_cs_ends(), st_cmp_cs_eq(), st_cmp_cs_starts(), stats_derived_count(), stats_derived_name(), stats_derived_value(), stats_get_count(), stats_get_name(), and stats_get_value_by_num().

Referenced by stats_derived_value().

5.136.2 Variable Documentation**5.136.2.1 size_t count**

Definition at line 11 of file statistics.c.

Referenced by __attribute__(), folder_count(), hashed_bucket(), hashed_cursor_active(), http_parse_pairs(), imap_copy(), imap_id(), imap_range_build(), imap_search_messages(), inx_count(), nvp_parse(), obj_cache_prune(), pop_get_message(), pop_stat(), portal_endpoint_alert_acknowledge(), portal_endpoint_folders_add(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(),

portal_endpoint_messages_list(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_message_attachments(), portal_message_tags_array(), portal_parse_flags(), portal_parse_json_str_array(), portal_parse_sections(), prime_unpack(), prime_unpack_fields(), prime_unpack_validate(), register_captcha_random_font(), res_row_next(), servers_get_count_using_port(), signature_tree_get(), signature_tree_verify(), slots_alloc(), slots_get(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stamp_counter_check(), str_tok_get_count_bl(), tank_count(), tok_get_count_bl(), and tree_truncate().

5.136.2.2 char* derived[]

Initial value:

```
{
    "system.secure.total",
    "system.secure.allocated",
    "system.secure.items",

    "core.spool.errors",
    "errors.total"
}
```

Definition at line 120 of file statistics.c.

Referenced by stats_derived_count(), and stats_derived_name().

5.136.2.3 pthread_mutex_t locks[128][2]

Definition at line 12 of file statistics.c.

5.136.2.4 char* names[128]

Definition at line 14 of file statistics.c.

5.136.2.5 struct { ... } stats

Referenced by portal_endpoint_folders_tags(), stats_adjust_by_name(), stats_adjust_by_num(), stats_decrement_by_name(), stats_decrement_by_num(), stats_get_count(), stats_get_name(), stats_get_name_pos(), stats_get_value_by_name(), stats_get_value_by_num(), stats_increment_by_name(), stats_increment_by_num(), stats_init(), stats_set_by_name(), stats_set_by_num(), and stats_shutdown().

5.136.2.6 uint64_t values[128]

Definition at line 13 of file statistics.c.

Referenced by imap_status().

5.137 src/web/statistics/statistics.c File Reference

```
#include "magma.h"
```

Functions

- void [statistics_process](#) ([connection_t](#) *con)

Display the statistics page to the requesting connection.

Variables

- [statistics_vp_t portal_stats](#) [12]

5.137.1 Function Documentation

5.137.1.1 void [statistics_process](#) ([connection_t](#) * con)

Display the statistics page to the requesting connection. [statistics.c](#)

Parameters:

con a pointer to the connection object across which the server statistics will be transmitted.

Returns:

This function returns no value.

Definition at line 17 of file [statistics.c](#).

References [con_write_st\(\)](#), [http_page_free\(\)](#), [http_page_get\(\)](#), [http_print_500\(\)](#), [http_response_header\(\)](#), [log_pedantic](#), [portal_stat_emails_received_today](#), [portal_stat_emails_received_week](#), [portal_stat_emails_sent_today](#), [portal_stat_emails_sent_week](#), [portal_stat_total_users](#), [portal_stat_users_checked_email_today](#), [portal_stat_users_checked_email_week](#), [portal_stat_users_registered_today](#), [portal_stat_users_registered_week](#), [portal_stat_users_sent_email_today](#), [portal_stat_users_sent_email_week](#), [st_free\(\)](#), [st_length_get\(\)](#), [statistics_refresh\(\)](#), [statistics_vp_t::val](#), [xml_dump_doc\(\)](#), [xml_set_xpath_ns\(\)](#), and [xml_set_xpath_uint64\(\)](#).

Referenced by [http_response\(\)](#).

5.137.2 Variable Documentation

5.137.2.1 [statistics_vp_t portal_stats](#)[12]

Definition at line 13 of file [datatier.c](#).

5.138 src/engine/status/status.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t status` (void)
Check to see if a worker thread should continue processing.
- void `status_set` (int value)
Set the status level to provided value.
- void `status_signal` (void)
Enqueued by the signal handler so the status can be updated asynchronously.
- int `status_get` (void)
Get the current status level.
- void `status_process` (void)
Update magma's basic process information (pid and start time).
- uint64_t `status_pid` (void)
Get the magma instance's pid.
- uint64_t `status_startup` (void)
Get the timestamp corresponding to when magma was started.

Variables

- struct {
 pid_t `pid`
 uint64_t `startup`
} `process`
- int `status_level` = 0
- pthread_rwlock_t `status_lock` = PTHREAD_RWLOCK_INITIALIZER

5.138.1 Function Documentation

5.138.1.1 `bool_t status` (void)

Check to see if a worker thread should continue processing. [status.c](#)

Note:

This function should be called periodically by worker threads.

Returns:

true if the caller should continue processing (status level is positive) or false otherwise.

Definition at line 26 of file `status.c`.

References `rwlock_lock_read()`, `rwlock_unlock()`, `status_level`, and `status_lock`.

5.138.1.2 `int status_get (void)`

Get the current status level.

See also:

[status\(\)](#)

Returns:

the current value of the status level.

Definition at line 66 of file status.c.

References `rwlock_lock_read()`, `rwlock_unlock()`, `status_level`, and `status_lock`.

5.138.1.3 `uint64_t status_pid (void)`

Get the magma instance's pid.

Returns:

the value of magma's pid.

Definition at line 88 of file status.c.

References `process`.

5.138.1.4 `void status_process (void)`

Update magma's basic process information (pid and start time).

Returns:

This function returns no value.

Definition at line 78 of file status.c.

References `process`, and `process_my_pid()`.

Referenced by `process_start()`, `system_fork_daemon()`, and `system_init_impersonation()`.

5.138.1.5 `void status_set (int value)`

Set the status level to provided value.

See also:

[status\(\)](#)

Parameters:

value the integer value of the new status level.

Returns:

This function returns no value.

Definition at line 43 of file status.c.

References `rwlock_lock_write()`, `rwlock_unlock()`, `status_level`, and `status_lock`.

Referenced by `main()`.

5.138.1.6 void status_signal (void)

Enqueued by the signal handler so the status can be updated asynchronously.

Definition at line 53 of file status.c.

References log_pedantic, rwlock_lock_write(), rwlock_unlock(), status_level, and status_lock.

Referenced by signal_shutdown().

5.138.1.7 uint64_t status_startup (void)

Get the timestamp corresponding to when magma was started.

Returns:

the UNIX-style date-time value when magma was started.

Definition at line 96 of file status.c.

References process.

5.138.2 Variable Documentation

5.138.2.1 pid_t pid

Definition at line 11 of file status.c.

Referenced by main(), process_find_pid(), process_kill(), and system_fork_daemon().

5.138.2.2 struct { ... } process

Referenced by status_pid(), status_process(), and status_startup().

5.138.2.3 uint64_t startup

Definition at line 12 of file status.c.

5.138.2.4 int status_level = 0

Definition at line 18 of file status.c.

Referenced by status(), status_get(), status_set(), and status_signal().

5.138.2.5 pthread_rwlock_t status_lock = PTHREAD_RWLOCK_INITIALIZER

Definition at line 19 of file status.c.

Referenced by status(), status_get(), status_set(), and status_signal().

5.139 src/engine/status/status.h File Reference

Functions

- void [stats_adjust_by_name](#) (char *name, int32_t value)
statistics.c
- void [stats_adjust_by_num](#) (uint64_t position, int32_t value)
Provided a statistic by index, increment its value by a specified amount.
- void [stats_decrement_by_name](#) (char *name)
Provided a statistic by name, decrement its value by 1.
- void [stats_decrement_by_num](#) (uint64_t position)
Provided a statistic by index, decrement its value by 1.
- uint64_t [stats_derived_count](#) (void)
Get the number of derived statistics that are tracked.
- char * [stats_derived_name](#) (uint64_t position)
Get the name of a derived statistic by index.
- uint64_t [stats_derived_value](#) (uint64_t position)
Get the value of a derived statistic by index.
- uint64_t [stats_get_count](#) (void)
Get the number of statistics being tracked.
- char * [stats_get_name](#) (uint64_t position)
Get the name of a statistic by its index.
- uint64_t [stats_get_name_pos](#) (char *name)
Get the index of a statistic by name.
- uint64_t [stats_get_value_by_name](#) (char *name)
Provided a statistic by name, get its value.
- uint64_t [stats_get_value_by_num](#) (uint64_t position)
Provided a statistic by index, get its value.
- void [stats_increment_by_name](#) (char *name)
Provided a statistic by name, increment its value by 1.
- void [stats_increment_by_num](#) (uint64_t position)
Provided a statistic by index, increment its value by 1.
- bool_t [stats_init](#) (void)
Initialize and reset all statistics counters.
- void [stats_set_by_name](#) (char *name, uint64_t value)
Provided a statistic by name, set its value.
- void [stats_set_by_num](#) (uint64_t position, uint64_t value)

Provided a statistic by index, set its value.

- void [stats_shutdown](#) (void)
Destroy all statistics locks.
- uint64_t [stats_sum_errors](#) (void)
Get the total sum of all error statistics maintained by magma (traditional and derived stats).
- uint64_t [perf_rdtsc](#) (void)
performance.c
- const char * [build_commit](#) (void)
build.c
- const char * [build_stamp](#) (void)
Get the magma build stamp.
- const char * [build_version](#) (void)
Get the magma version string.
- uint64_t [build_version_major](#) (void)
Get the magma major version number.
- uint64_t [build_version_minor](#) (void)
Get the magma minor version number.
- uint64_t [build_version_patch](#) (void)
Get the magma patch level.
- [bool_t status](#) (void)
status.c
- int [status_get](#) (void)
Get the current status level.
- uint64_t [status_pid](#) (void)
Get the magma instance's pid.
- void [status_process](#) (void)
Update magma's basic process information (pid and start time).
- void [status_set](#) (int value)
Set the status level to provided value.
- void [status_signal](#) (void)
Enqueued by the signal handler so the status can be updated asynchronously.
- uint64_t [status_startup](#) (void)
Get the timestamp corresponding to when magma was started.

5.139.1 Function Documentation

5.139.1.1 `const char* build_commit (void)`

[build.c](#) [build.c](#)

Returns:

a pointer to a null-terminated string containing the last eight characters of current commit identity.

Definition at line 34 of file `build.c`.

Referenced by `args_parse()`, and `lib_load()`.

5.139.1.2 `const char* build_stamp (void)`

Get the magma build stamp.

Returns:

a pointer to a null-terminated string containing the magma build information string.

Definition at line 42 of file `build.c`.

Referenced by `args_parse()`, `http_response_options()`, `imap_id()`, and `lib_load()`.

5.139.1.3 `const char* build_version (void)`

Get the magma version string.

Returns:

a pointer to a null-terminated string containing the magma version string.

Definition at line 26 of file `build.c`.

Referenced by `args_parse()`, `http_response_options()`, `imap_id()`, `imap_init()`, `lib_load()`, and `pop_capa()`.

5.139.1.4 `uint64_t build_version_major (void)`

Get the magma major version number.

Returns:

an unsigned integer with the major version number.

Definition at line 50 of file `build.c`.

References `ns_length_get()`, `pl_null()`, `PLACER`, `placer_t`, `tok_get_st()`, and `uint64_conv_pl()`.

5.139.1.5 `uint64_t build_version_minor (void)`

Get the magma minor version number.

Returns:

an unsigned integer with the minor version number.

Definition at line 66 of file `build.c`.

References `ns_length_get()`, `pl_null()`, `PLACER`, `placer_t`, `tok_get_st()`, and `uint64_conv_pl()`.

5.139.1.6 uint64_t build_version_patch (void)

Get the magma patch level.

Returns:

an unsigned integer with the patch level.

Definition at line 82 of file build.c.

References ns_length_get(), pl_null(), PLACER, placer_t, tok_get_st(), and uint64_conv_pl().

5.139.1.7 uint64_t perf_rdtsc (void)

[performance.c](#) Returns the CPU cycle counter. By calculating variations in the time stamp, we can discover the exact number of elapsed CPU ticks, making it possible to precisely time code execution. Note that in a multithreaded environment its possible that not all of the elapsed ticks were consumed by the code being timed.

Returns:

The number of CPU cycles that have elapsed since boot.

Definition at line 180 of file performance.c.

5.139.1.8 void stats_adjust_by_name (char * name, int32_t value)

statistics.c statistics.c

Parameters:

name a null-terminated string containing the name of the statistic to be set.

value the value by which to increment the specified statistic.

Returns:

This function returns no value.

Definition at line 360 of file statistics.c.

References mutex_lock(), mutex_unlock(), stats, and stats_get_name_pos().

Referenced by dkim_signature_create(), dkim_signature_verify(), obj_cache_prune(), pattern_check(), and spf_check().

5.139.1.9 void stats_adjust_by_num (uint64_t position, int32_t value)

Provided a statistic by index, increment its value by a specified amount.

Parameters:

position the zero-based index of the statistic to be set.

value the value by which to increment the specified statistic.

Returns:

This function returns no value.

Definition at line 381 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

5.139.1.10 void stats_decrement_by_name (char * *name*)

Provided a statistic by name, decrement its value by 1.

Parameters:

name a null-terminated string containing the name of the statistic to be decremented.

Returns:

This function returns no value.

Definition at line 429 of file statistics.c.

References mutex_lock(), mutex_unlock(), stats, and stats_get_name_pos().

Referenced by con_destroy(), dequeue(), and queue_shutdown().

5.139.1.11 void stats_decrement_by_num (uint64_t *position*)

Provided a statistic by index, decrement its value by 1.

Parameters:

position the zero-based index of the statistic to be decremented.

Returns:

This function returns no value.

Definition at line 449 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

5.139.1.12 uint64_t stats_derived_count (void)

Get the number of derived statistics that are tracked.

Returns:

the number of derived statistics being maintained by magma.

Definition at line 183 of file statistics.c.

References derived.

Referenced by molten_stats(), stats_derived_name(), and stats_sum_errors().

5.139.1.13 char* stats_derived_name (uint64_t *position*)

Get the name of a derived statistic by index.

Parameters:

position the zero-based index of the derived statistic to be queried.

Returns:

NULL on failure, or a pointer to a null-terminated string containing the name of the derived statistic on success.

Definition at line 193 of file statistics.c.

References derived, and stats_derived_count().

Referenced by molten_stats(), and stats_sum_errors().

5.139.1.14 uint64_t stats_derived_value (uint64_t *position*)

Get the value of a derived statistic by index.

See also:

[mm_sec_stats\(\)](#)

Parameters:

position the zero-based index of the derived statistic to be queried.

Returns:

0 on failure, or the value of the specified derived statistic on success.

Definition at line 208 of file statistics.c.

References bytes, items, log_pedantic, mm_sec_stats(), spool_error_stats(), and stats_sum_errors().

Referenced by molten_stats(), and stats_sum_errors().

5.139.1.15 uint64_t stats_get_count (void)

Get the number of statistics being tracked.

Returns:

the total number of statistics counters maintained by magma.

Definition at line 462 of file statistics.c.

References stats.

Referenced by molten_stats(), and stats_sum_errors().

5.139.1.16 char* stats_get_name (uint64_t *position*)

Get the name of a statistic by its index.

Parameters:

position the zero-based index of the statistic to be queried.

Returns:

0 on failure, or the name of the requested statistic on success.

Definition at line 272 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

Referenced by molten_stats(), and stats_sum_errors().

5.139.1.17 uint64_t stats_get_name_pos (char * *name*)

Get the index of a statistic by name.

Parameters:

name the name of the statistic to be queried.

Returns:

0 on failure, or the zero-based index of the requested statistic on success.

Definition at line 251 of file statistics.c.

References log_info, mutex_lock(), mutex_unlock(), NULLER, st_cmp_cs_eq(), and stats.

Referenced by stats_adjust_by_name(), stats_decrement_by_name(), stats_get_value_by_name(), stats_increment_by_name(), and stats_set_by_name().

5.139.1.18 uint64_t stats_get_value_by_name (char * name)

Provided a statistic by name, get its value.

Parameters:

name a null-terminated string containing the name of the statistic to be queried.

Returns:

the specified statistic's value, as an unsigned 64 bit integer.

Definition at line 323 of file statistics.c.

References mutex_lock(), mutex_unlock(), stats, and stats_get_name_pos().

5.139.1.19 uint64_t stats_get_value_by_num (uint64_t position)

Provided a statistic by index, get its value.

Parameters:

position the zero-based index of the statistic to be queried.

Returns:

the specified statistic's value, as an unsigned 64 bit integer.

Definition at line 343 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

Referenced by molten_stats(), and stats_sum_errors().

5.139.1.20 void stats_increment_by_name (char * name)

Provided a statistic by name, increment its value by 1.

Parameters:

name a null-terminated string containing the name of the statistic to be incremented.

Returns:

This function returns no value.

Definition at line 395 of file statistics.c.

References mutex_lock(), mutex_unlock(), stats, and stats_get_name_pos().

Referenced by dequeue(), imap_starttls(), pop_starttls(), protocol_enqueue(), queue_init(), register_abuse_check_blocklist(), smtp_starttls(), virus_check(), virus_engine_refresh(), and virus_start().

5.139.1.21 void stats_increment_by_num (uint64_t *position*)

Provided a statistic by index, increment its value by 1.

Parameters:

position the zero-based index of the statistic to be incremented.

Returns:

This function returns no value.

Definition at line 415 of file statistics.c.

References mutex_lock(), mutex_unlock(), and stats.

5.139.1.22 bool_t stats_init (void)

Initialize and reset all statistics counters.

Returns:

false on failure or true on success.

Definition at line 471 of file statistics.c.

References log_critical, mm_wipe(), mutex_init(), and stats.

Referenced by process_start().

5.139.1.23 void stats_set_by_name (char * *name*, uint64_t *value*)

Provided a statistic by name, set its value.

Parameters:

name a null-terminated string containing the name of the statistic to be set.

value the new value of the specified statistic.

Returns:

This function returns no value.

Definition at line 289 of file statistics.c.

References mutex_lock(), mutex_unlock(), stats, and stats_get_name_pos().

Referenced by obj_cache_prune(), virus_engine_refresh(), virus_start(), and virus_stop().

5.139.1.24 void stats_set_by_num (uint64_t *position*, uint64_t *value*)

Provided a statistic by index, set its value.

Parameters:

position the zero-based index of the statistic to be set.

value the new value of the specified statistic.

Returns:

This function returns no value.

Definition at line 309 of file statistics.c.

References `mutex_lock()`, `mutex_unlock()`, and `stats`.

5.139.1.25 `void stats_shutdown (void)`

Destroy all statistics locks.

Returns:

This function returns no value.

Definition at line 494 of file statistics.c.

References `mutex_destroy()`, and `stats`.

Referenced by `process_stop()`.

5.139.1.26 `uint64_t stats_sum_errors (void)`

Get the total sum of all error statistics maintained by magma (traditional and derived stats).

Note:

Error statistics are all statistics that have a name that begins with "errors." or ends with ".errors".

Returns:

The total sum of all error statistics counters accumulated by magma.

Definition at line 137 of file statistics.c.

References `CONSTANT`, `NULLER`, `st_cmp_cs_ends()`, `st_cmp_cs_eq()`, `st_cmp_cs_starts()`, `stats_derived_count()`, `stats_derived_name()`, `stats_derived_value()`, `stats_get_count()`, `stats_get_name()`, and `stats_get_value_by_num()`.

Referenced by `stats_derived_value()`.

5.139.1.27 `bool_t status (void)`

[status.c](#) [status.c](#)

Note:

This function should be called periodically by worker threads.

Returns:

true if the caller should continue processing (status level is positive) or false otherwise.

Definition at line 26 of file status.c.

References `rwlock_lock_read()`, `rwlock_unlock()`, `status_level`, and `status_lock`.

5.139.1.28 `int status_get (void)`

Get the current status level.

See also:

[status\(\)](#)

Returns:

the current value of the status level.

Definition at line 66 of file status.c.

References `rwlock_lock_read()`, `rwlock_unlock()`, `status_level`, and `status_lock`.

5.139.1.29 uint64_t status_pid (void)

Get the magma instance's pid.

Returns:

the value of magma's pid.

Definition at line 88 of file status.c.

References `process`.

5.139.1.30 void status_process (void)

Update magma's basic process information (pid and start time).

Returns:

This function returns no value.

Definition at line 78 of file status.c.

References `process`, and `process_my_pid()`.

Referenced by `process_start()`, `system_fork_daemon()`, and `system_init_impersonation()`.

5.139.1.31 void status_set (int value)

Set the status level to provided value.

See also:

[status\(\)](#)

Parameters:

value the integer value of the new status level.

Returns:

This function returns no value.

Definition at line 43 of file status.c.

References `rwlock_lock_write()`, `rwlock_unlock()`, `status_level`, and `status_lock`.

Referenced by `main()`.

5.139.1.32 void status_signal (void)

Enqueued by the signal handler so the status can be updated asynchronously.

Definition at line 53 of file status.c.

References `log_pedantic`, `rwlock_lock_write()`, `rwlock_unlock()`, `status_level`, and `status_lock`.

Referenced by `signal_shutdown()`.

5.139.1.33 `uint64_t status_startup (void)`

Get the timestamp corresponding to when magma was started.

Returns:

the UNIX-style date-time value when magma was started.

Definition at line 96 of file `status.c`.

References `process`.

5.140 src/magma.c File Reference

```
#include "magma.h"
```

Functions

- int [main](#) (int argc, char *argv[])

Variables

- FILE * [log_descriptor](#)

5.140.1 Function Documentation

5.140.1.1 int main (int *argc*, char * *argv*[])

Definition at line 12 of file magma.c.

References [args_parse\(\)](#), [log_descriptor](#), [log_error](#), [net_listen\(\)](#), [pid](#), [PLACER](#), [process_find_pid\(\)](#), [process_start\(\)](#), [process_stop\(\)](#), and [status_set\(\)](#).

5.140.2 Variable Documentation

5.140.2.1 FILE* [log_descriptor](#)

Definition at line 12 of file log.c.

Referenced by [log_internal\(\)](#), [log_rotate\(\)](#), [log_start\(\)](#), and [main\(\)](#).

5.141 src/magma.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <unistd.h>
#include <stddef.h>
#include <limits.h>
#include <signal.h>
#include <string.h>
#include <dirent.h>
#include <pwd.h>
#include <errno.h>
#include <fcntl.h>
#include <inttypes.h>
#include <pthread.h>
#include <stdarg.h>
#include <dlfcn.h>
#include <execinfo.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/resource.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <sys/utsname.h>
#include <sys/prctl.h>
#include <sys/epoll.h>
#include <sys/sysctl.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#include <netdb.h>
#include <resolv.h>
#include <regex.h>
#include <ftw.h>
#include <search.h>
#include <semaphore.h>
#include <sys/mman.h>
```



```
#include <gnu/libc-version.h>
#include "providers/symbols.h"
#include "core/core.h"
#include "providers/providers.h"
#include "engine/engine.h"
#include "network/network.h"
#include "objects/objects.h"
#include "servers/servers.h"
#include "web/web.h"
#include "queries.h"
```

Defines

- #define [bufptr](#) (char *)&(threadBuffer)
- #define [buflen](#) sizeof(threadBuffer)

Variables

- [magma_t](#) magma
- __thread char [threadBuffer](#) [1024]

5.141.1 Define Documentation

5.141.1.1 #define buflen sizeof(threadBuffer)

Definition at line 72 of file magma.h.

Referenced by [con_print\(\)](#), [net_set_blocking\(\)](#), [net_set_buffer_length\(\)](#), [net_set_keepalive\(\)](#), [net_set_linger\(\)](#), [net_set_nodelay\(\)](#), [net_set_reuseable_address\(\)](#), [net_set_timeout\(\)](#), [rand_get_int16\(\)](#), [rand_get_int32\(\)](#), [rand_get_int64\(\)](#), [rand_get_int8\(\)](#), [rand_get_uint16\(\)](#), [rand_get_uint32\(\)](#), [rand_get_uint64\(\)](#), [rand_get_uint8\(\)](#), [sdscatvprintf\(\)](#), [servers_validate\(\)](#), [system_change_root_directory\(\)](#), [system_init_core_dumps\(\)](#), [system_init_impersonation\(\)](#), [system_init_umask\(\)](#), [system_ulimit_cur\(\)](#), and [system_ulimit_max\(\)](#).

5.141.1.2 #define bufptr (char *)&(threadBuffer)

Definition at line 71 of file magma.h.

Referenced by [_compute_sha_hash_multibuf\(\)](#), [_deserialize_ec_privkey\(\)](#), [con_print\(\)](#), [net_set_blocking\(\)](#), [net_set_buffer_length\(\)](#), [net_set_keepalive\(\)](#), [net_set_linger\(\)](#), [net_set_nodelay\(\)](#), [net_set_reuseable_address\(\)](#), [net_set_timeout\(\)](#), [rand_get_int16\(\)](#), [rand_get_int32\(\)](#), [rand_get_int64\(\)](#), [rand_get_int8\(\)](#), [rand_get_uint16\(\)](#), [rand_get_uint32\(\)](#), [rand_get_uint64\(\)](#), [rand_get_uint8\(\)](#), [servers_validate\(\)](#), [system_change_root_directory\(\)](#), [system_init_core_dumps\(\)](#), [system_init_impersonation\(\)](#), [system_init_umask\(\)](#), [system_ulimit_cur\(\)](#), and [system_ulimit_max\(\)](#).

5.141.2 Variable Documentation

5.141.2.1 magma_t magma

Definition at line 12 of file global.c.

Referenced by [api_response\(\)](#), [args_parse\(\)](#), [auth_legacy\(\)](#), [auth_login\(\)](#), [auth_sanitize_username\(\)](#), [cache_alloc\(\)](#), [cache_free\(\)](#), [cache_output_settings\(\)](#), [cache_start\(\)](#), [cache_stop\(\)](#), [cache_validate\(\)](#), [color_supported\(\)](#), [con_init_network_buffer\(\)](#), [config_fetch_host_number\(\)](#),

config_fetch_settings(), contact_business(), dh_params_generate(), dkim_signature_create(), dkim_start(), http_body(), http_content_load_fonts(), http_content_refresh(), http_content_start(), http_data_value_parse(), http_load_file(), http_parse_header(), http_parse_method(), http_print_301(), http_response(), http_response_connection(), http_response_header(), http_response_options(), imap_append_message(), imap_command_parser(), jansson_flags(), json_api_dispatch(), lib_load(), lib_unload(), log_internal(), log_rotate(), log_start(), mail_add_forward_headers(), mail_add_outbound_headers(), mail_create_directory(), mail_db_insert_duplicate_message(), mail_db_insert_message(), mail_message_cleanup(), mail_message_path(), mm_sec_start(), net_init(), net_listen(), net_trigger(), portal_endpoint(), portal_endpoint_error(), portal_endpoint_response(), portal_process(), portal_upload(), prime_start(), process_start(), process_stop(), queue_init(), queue_shutdown(), queue_signal(), rand_start(), rand_thread_start(), register_business_step1(), register_business_step2(), register_business_validate_password(), register_captcha_random_font(), register_data_insert_user(), relay_alloc(), relay_counter(), relay_free(), relay_output_settings(), relay_validate(), servers_alloc(), servers_encryption_start(), servers_encryption_stop(), servers_free(), servers_get_by_protocol(), servers_get_by_socket(), servers_get_count_using_port(), servers_network_start(), servers_network_stop(), servers_output_settings(), servers_validate(), sess_create(), sess_get(), sess_token(), smtp_accept_message(), smtp_add_bypass_entry(), smtp_bounce(), smtp_bypass_check(), smtp_check_rbl(), smtp_client_connect(), smtp_client_send_helo(), smtp_data(), smtp_ehlo(), smtp_mail_from(), smtp_parse_helo_domain(), smtp_parse_mail_from_path(), smtp_parse_rcpt_to(), smtp_rcpt_to(), spf_start(), spf_stop(), spool_path(), sql_open(), sql_start(), sql_stop(), st_alloc_opts(), st_realloc(), stmt_start(), stmt_stop(), system_change_root_directory(), system_fork_daemon(), system_init_core_dumps(), system_init_impersonation(), system_init_resource_limits(), tank_start(), thread_launch(), tls_server_create(), virus_check(), virus_engine_create(), virus_engine_refresh(), virus_sigs_total(), virus_start(), and virus_stop().

5.141.2.2 __thread char threadBuffer[1024]

Definition at line 11 of file global.c.

5.142 src/network/addresses.c File Reference

```
#include "magma.h"
```

Functions

- `ip_t * con_addr (connection_t *con, ip_t *output)`
Return the IP address information of the remote end of a client connection.
- `stringer_t * con_addr_presentation (connection_t *con, stringer_t *output)`
Return a textual representation of the IP address of a specified connection handle.
- `stringer_t * con_addr_standard (connection_t *con, stringer_t *output)`
Get the IP address string for the client connection.
- `stringer_t * con_addr_reversed (connection_t *con, stringer_t *output)`
Get the reversed-IP address string for a client connection.
- `stringer_t * con_addr_subnet (connection_t *con, stringer_t *output)`
Get the subnet string for a specified connection.
- `uint32_t con_addr_word (connection_t *con, int_t position)`
Get a specified 32-bit segment of a connection's peer IP address.
- `octet_t con_addr_octet (connection_t *con, int_t position)`
Get a specified 8-bit octet of a connection's peer IP address.
- `segment_t con_addr_segment (connection_t *con, int_t position)`
Extract a specified 16 bit segment from a connection's peer IP address.

5.142.1 Function Documentation

5.142.1.1 ip_t* con_addr (connection_t * con, ip_t * output)

Return the IP address information of the remote end of a client connection. [addresses.c](#)

Parameters:

con the input client connection.

output a pointer to an `ip_t` structure to receive the remote IP address, which is allocated for the caller if output is NULL.

Returns:

NULL on error, or a pointer to the IP address of the remote host.

Definition at line 16 of file `addresses.c`.

References `ip_t::ip`.

Referenced by `con_addr_octet()`, `con_addr_presentation()`, `con_addr_reversed()`, `con_addr_segment()`, `con_addr_standard()`, `con_addr_subnet()`, `con_addr_word()`, `sess_create()`, `sess_get()`, `smtp_bypass_check()`, and `smtp_rcpt_to()`.

5.142.1.2 `octet_t con_addr_octet (connection_t * con, int_t position)`

Get a specified 8-bit octet of a connection's peer IP address.

Parameters:

con a pointer to the connection object to be examined.

position a zero-based index into the 8-bit octets that comprise the target IP address.

Returns:

-1 on error, or the specified 8-bit octet of the passed address on success.

Definition at line 152 of file addresses.c.

References `con_addr()`, and `ip_octet()`.

5.142.1.3 `stringer_t* con_addr_presentation (connection_t * con, stringer_t * output)`

Return a textual representation of the IP address of a specified connection handle.

Parameters:

con the remote connection to be queried.

output a managed string to store the result, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to a managed string containing a textual representation of the IP address.

Definition at line 62 of file addresses.c.

References `con_addr()`, and `ip_presentation()`.

Referenced by `api_endpoint_auth()`, `contact_abuse_checks()`, `contact_abuse_increment_history()`, `contact_business()`, `imap_id()`, `imap_login()`, `mail_add_inbound_headers()`, `mail_add_outbound_headers()`, `pop_pass()`, `portal_endpoint_auth()`, `portal_meta()`, `protocol_secure()`, `register_abuse_checks()`, `register_data_insert_user()`, `register_session_cache()`, `register_session_get()`, `smtp_auth_login()`, `smtp_auth_plain()`, and `smtp_rcpt_to()`.

5.142.1.4 `stringer_t* con_addr_reversed (connection_t * con, stringer_t * output)`

Get the reversed-IP address string for a client connection.

Parameters:

con the client connection to be queried.

output a managed string to receive the output, which will be allocated for the caller if passed as NULL.

Returns:

NULL on failure, or a pointer to a managed string containing a textual representation of the IP address on success.

Definition at line 98 of file addresses.c.

References `con_addr()`, and `ip_reversed()`.

Referenced by `smtp_check_greylist()`, and `smtp_check_rbl()`.

5.142.1.5 `segment_t con_addr_segment(connection_t * con, int_t position)`

Extract a specified 16 bit segment from a connection's peer IP address.

Parameters:

con a pointer to the connection object to be examined.

position the zero-indexed (starting at least-significant word) 16-bit segment of the IP address to be evaluated.

Returns:

-1 on failure, or the 32 bit-widened segment extracted from the supplied IP address.

Definition at line 170 of file addresses.c.

References `con_addr()`, and `ip_segment()`.

5.142.1.6 `stringer_t* con_addr_standard(connection_t * con, stringer_t * output)`

Get the IP address string for the client connection.

Parameters:

con the client connection to be queried.

output a managed string to receive the output, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to a managed string containing a textual representation of the IP address.

Definition at line 80 of file addresses.c.

References `con_addr()`, and `ip_standard()`.

Referenced by `register_abuse_check_blocklist()`, `register_abuse_checks()`, and `register_abuse_increment_history()`.

5.142.1.7 `stringer_t* con_addr_subnet(connection_t * con, stringer_t * output)`

Get the subnet string for a specified connection.

Parameters:

con the client connection to be queried.

output a managed string that will store the output of the subnet string lookup.

Returns:

NULL on failure, or a managed string containing a textual representation of a subnet address on success.

Definition at line 116 of file addresses.c.

References `con_addr()`, and `ip_subnet()`.

Referenced by `api_endpoint_auth()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_check_receive_quota()`, and `smtp_update_receive_stats()`.

5.142.1.8 uint32_t con_addr_word (connection_t * *con*, int_t *position*)

Get a specified 32-bit segment of a connection's peer IP address.

Parameters:

con a pointer to the connection object to be examined.

position a zero-based index into the 32-bit word(s) that comprise the target IP address.

Returns:

-1 if the connecting address can't be looked up, 0 on general error, or the specified 32-bit segment of the passed address on success.

Definition at line 134 of file addresses.c.

References `con_addr()`, and `ip_word()`.

5.143 src/network/clients.c File Reference

```
#include "magma.h"
```

Functions

- [int_t client_status](#) ([client_t](#) *client)
Get the status of a network client.
- [int_t client_secure](#) ([client_t](#) *client)
Establish a TLS connection with a network client instance.
- [client_t * client_connect](#) ([chr_t](#) *host, uint32_t port)
Establish a network client connection to a remote host.
- void [client_close](#) ([client_t](#) *client)
Close a network client connection.

5.143.1 Function Documentation

5.143.1.1 void client_close (client_t * client)

Close a network client connection. [clients.c](#)

Note:

If ssl is in use, this will also destroy the overlying ssl session.

Returns:

This function returns no value.

Definition at line 175 of file clients.c.

References [mm_cleanup](#), [mm_free\(\)](#), [st_cleanup](#), and [tls_free\(\)](#).

Referenced by [net_trigger\(\)](#), [smtp_client_close\(\)](#), and [smtp_client_connect\(\)](#).

5.143.1.2 client_t* client_connect (chr_t * host, uint32_t port)

Establish a network client connection to a remote host.

Parameters:

host a pointer to a null-terminated string containing the hostname of the remote server.

port the port number of the server to which the connection will be established.

Returns:

NULL on failure or a pointer to a newly initialized network client object for the connection upon success.

Definition at line 70 of file clients.c.

References [client_t](#), [ip_t::family](#), [FOREIGNDATA](#), [ip_t::ip4](#), [ip_t::ip6](#), [JOINTED](#), [log_pedantic](#), [MEMORYBUF](#), [mm_alloc\(\)](#), [mm_cleanup](#), [mm_copy\(\)](#), [mm_free\(\)](#), [PLACER_T](#), [st_alloc\(\)](#), [STACK](#), and [status](#).

Referenced by [net_trigger\(\)](#), and [smtp_client_connect\(\)](#).

5.143.1.3 `int_t client_secure (client_t * client)`

Establish a TLS connection with a network client instance.

Parameters:

client a pointer to the network client object to have its transport security upgraded.

Returns:

-1 on failure or 0 on success.

Definition at line 45 of file clients.c.

References `tls_client_alloc()`.

Referenced by `smtp_client_connect()`.

5.143.1.4 `int_t client_status (client_t * client)`

Get the status of a network client.

Parameters:

client a pointer to the network client object to be queried.

Returns:

-1 on network errors, 0 for an unknown status, 1 for connected, and 2 for a graceful shutdown.

Definition at line 15 of file clients.c.

References `tcp_status()`, and `tls_status()`.

Referenced by `client_read()`, and `client_write()`.

5.144 src/network/connections.c File Reference

```
#include "magma.h"
```

Functions

- [int_t con_secure](#) ([connection_t](#) *con)
Return the security level of a specified connection.
- [bool_t con_localhost](#) ([connection_t](#) *con)
Determine whether a client connection is from the same machine, using the loopback adapter, or a remote machine.
- [bool_t con_private](#) ([connection_t](#) *con)
Determine whether a client connection is with a machine on the same private network.
- [int_t con_status](#) ([connection_t](#) *con)
Return the status of a specified connection.
- void [con_flush](#) ([connection_t](#) *con)
Attempt to flush any buffered data associated with a network connection.
- void [con_destroy](#) ([connection_t](#) *con)
Destroy and free a generic connection object after executing its protocol-specific destructor; update any statistics accordingly.
- [uint64_t con_increment_refs](#) ([connection_t](#) *con)
Increment a connection object's reference counter.
- [uint64_t con_decrement_refs](#) ([connection_t](#) *con)
Decrement a connection object's reference counter.
- [bool_t con_init_network_buffer](#) ([connection_t](#) *con)
Allocate and initialize a new network buffer for a connection, if it is not already associated with one.
- [connection_t](#) * [con_init](#) (int cond, [server_t](#) *server)
Create a new connection object for a client connection.

5.144.1 Function Documentation

5.144.1.1 [uint64_t con_decrement_refs](#) ([connection_t](#) * con)

Decrement a connection object's reference counter. [connections.c](#)

Parameters:

con the connection object being referenced.

Returns:

an integer containing the new number of references to the specified connection.

Definition at line 210 of file [connections.c](#).

References [mutex_lock\(\)](#), and [mutex_unlock\(\)](#).

Referenced by [con_destroy\(\)](#).

5.144.1.2 void con_destroy (connection_t * con)

Destroy and free a generic connection object after executing its protocol-specific destructor; update any statistics accordingly.

Parameters:

con a pointer to the connection to be destroyed.

Returns:

This function returns no value.

Definition at line 104 of file connections.c.

References con_decrement_refs(), DMTP, dmtplib_session_destroy(), HTTP, http_session_destroy(), IMAP, imap_session_destroy(), mm_cleanup, mm_free(), MOLTEN, molten_session_destroy(), mutex_destroy(), POP, pop_session_destroy(), SMTP, smtp_session_destroy(), st_cleanup, stats_decrement_by_name(), SUBMISSION, and tls_free().

Referenced by con_reverse_lookup(), dmtplib_quit(), http_close(), imap_logout(), molten_quit(), pop_quit(), protocol_enqueue(), and smtp_quit().

5.144.1.3 void con_flush (connection_t * con)

Attempt to flush any buffered data associated with a network connection.

Parameters:

con the input client connection.

Returns:

This function returns no value.

Definition at line 89 of file connections.c.

References net_set_nodelay().

Referenced by smtp_quit().

5.144.1.4 uint64_t con_increment_refs (connection_t * con)

Increment a connection object's reference counter.

Parameters:

con the connection object being referenced.

Returns:

an integer containing the new number of references to the specified connection.

Definition at line 194 of file connections.c.

References mutex_lock(), and mutex_unlock().

Referenced by con_init().

5.144.1.5 connection_t* con_init (int cond, server_t * server)

Create a new connection object for a client connection.

Parameters:

cond the socket descriptor of the inbound client connection that was just accepted.

server the handle of the server that serviced the inbound request.

Definition at line 250 of file connections.c.

References `con_increment_refs()`, `connection_t`, `mm_alloc()`, `mm_free()`, `mutex_init()`, `server_t::network`, `server_t::sockd`, and `tcp_addr_ip()`.

Referenced by `protocol_process()`.

5.144.1.6 bool_t con_init_network_buffer (connection_t * con)

Allocate and initialize a new network buffer for a connection, if it is not already associated with one.

Note:

A network buffer of size `magma.system.network_buffer` bytes will be allocated for the connection if one does not exist.

Returns:

true if the connection object has been associated with a network buffer or false on failure.

Definition at line 226 of file connections.c.

References `log_info`, `magma`, `magma_t::network_buffer`, `pl_null()`, `st_alloc()`, and `magma_t::system`.

Referenced by `con_read()`, and `con_read_line()`.

5.144.1.7 bool_t con_localhost (connection_t * con)

Determine whether a client connection is from the same machine, using the loopback adapter, or a remote machine.

See also:

[ip_localhost\(\)](#)

Returns:

true if the connection appears to be using the loopback adapter, or false, if the connection appears to be from anywhere else.

Definition at line 32 of file connections.c.

References `ip_localhost()`.

Referenced by `json_api_dispatch()`, `portal_endpoint()`, `portal_endpoint_auth()`, `portal_process()`, and `portal_upload()`.

5.144.1.8 bool_t con_private (connection_t * con)

Determine whether a client connection is with a machine on the same private network.

See also:

[ip_private\(\)](#)

Returns:

true if the connection appears to be using the loopback adapter, or false, if the connection appears to be from anywhere else.

Definition at line 48 of file connections.c.

References `ip_localhost()`.

5.144.1.9 `int_t con_secure (connection_t * con)`

Return the security level of a specified connection.

Parameters:

con the input client connection.

Returns:

1 for a secure connection, 0 for insecure, and -1 if the server does not support SSL/TLS.

Definition at line 15 of file connections.c.

Referenced by `contact_process()`, `http_print_301()`, `http_response_cookie()`, `imap_capability()`, `imap_init()`, `imap_login()`, `imap_starttls()`, `json_api_dispatch()`, `pop_capa()`, `pop_pass()`, `pop_starttls()`, `portal_endpoint()`, `portal_endpoint_auth()`, `portal_process()`, `portal_upload()`, `protocol_enqueue()`, `register_process()`, `sess_create()`, `sess_get()`, `smtp_ehlo()`, `smtp_rcpt_to()`, `smtp_starttls()`, and `teacher_process()`.

5.144.1.10 `int_t con_status (connection_t * con)`

Return the status of a specified connection.

Parameters:

con the input client connection.

Returns:

-1 on network errors, 0 for an unknown status, 1 for connected, and 2 for a graceful shutdown.

Definition at line 64 of file connections.c.

References `tcp_status()`, and `tls_status()`.

Referenced by `con_read()`, `con_read_line()`, `con_write_bl()`, `dmtip_quit()`, `dmtip_requeue()`, `http_requeue()`, `imap_fetch()`, `imap_logout()`, `imap_requeue()`, `pop_quit()`, `pop_requeue()`, `smtp_quit()`, and `smtp_requeue()`.

5.145 src/network/dmtp.h File Reference

Data Structures

- struct [dmtp_session_t](#)

5.146 src/providers/dime/signet-resolver/dmtp.h File Reference

```
#include "dime/signet/signet.h"
#include "dime/signet-resolver/mrec.h"
#include "dime/common/error.h"
#include "dime/signet-resolver/signet-ssl.h"
```

Data Structures

- struct [dmtp_session_t](#)

Defines

- #define [DMTP_PORT](#) 26
- #define [DMTP_PORT_DUAL](#) 25
- #define [DMTP_V1_CIPHER_LIST](#) "ECDHE-RSA-AES256-GCM-SHA384"
- #define [DMTP_MAX_MX_RETRIES](#) 3
- #define [DMTP_LINE_BUF_SIZE](#) 4096

Enumerations

- enum [dmtp_mode_t](#) {
[DMTP_MODE_DMTP](#), [DMTP_MODE_SMTP](#), [DMTP_MODE_NONE](#), [dmtp_mode_unknown](#) = 0,
[dmtp_mode_dual](#) = 1, [dmtp_mode_dmtp](#) = 2, [dmtp_mode_smtp](#) = 3, [dmtp_mode_esmtp](#) = 4 }
- enum [dmtp_mail_rettype_t](#) { [return_type_default](#) = 0, [return_type_full](#) = 1, [return_type_display](#) = 2, [return_type_header](#) = 3 }
- enum [dmtp_mail_datatype_t](#) { [data_type_default](#) = 0, [data_type_7bit](#) = 1, [data_type_8bit](#) = 2 }

Functions

- [PUBLIC_FUNC_DECL](#) ([signet_t](#) *, [get_signet](#), const char *name, const char *fingerprint, int use_cache)
- [PUBLIC_FUNC_DECL](#) ([dmtp_session_t](#) *, [sgnt_resolv_dmtp_connect](#), const char *domain, int force_family)
- [PUBLIC_FUNC_DECL](#) (void, [sgnt_resolv_destroy_dmtp_session](#), [dmtp_session_t](#) *session)
- [PUBLIC_FUNC_DECL](#) ([dmtp_session_t](#) *, [dx_connect_standard](#), const char *host, const char *domain, int force_family, [dime_record_t](#) *dimerec)
- [PUBLIC_FUNC_DECL](#) ([dmtp_session_t](#) *, [dx_connect_dual](#), const char *host, const char *domain, int force_family, [dime_record_t](#) *dimerec, int failover)
- [PUBLIC_FUNC_DECL](#) (int, [verify_dx_certificate](#), [dmtp_session_t](#) *session)
- [PUBLIC_FUNC_DECL](#) (int, [sgnt_resolv_dmtp_ehlo](#), [dmtp_session_t](#) *session, const char *domain)
- [PUBLIC_FUNC_DECL](#) (int, [sgnt_resolv_dmtp_mail_from](#), [dmtp_session_t](#) *session, const char *origin, size_t msgsize, [dmtp_mail_rettype_t](#) rettype, [dmtp_mail_datatype_t](#) dtype)
- [PUBLIC_FUNC_DECL](#) (int, [sgnt_resolv_dmtp_rcpt_to](#), [dmtp_session_t](#) *session, const char *domain)
- [PUBLIC_FUNC_DECL](#) (char *, [sgnt_resolv_dmtp_data](#), [dmtp_session_t](#) *session, void *msg, size_t msglen)
- [PUBLIC_FUNC_DECL](#) (char *, [sgnt_resolv_dmtp_get_signet](#), [dmtp_session_t](#) *session, const char *signame, const char *fingerprint)
- [PUBLIC_FUNC_DECL](#) (int, [sgnt_resolv_dmtp_verify_signet](#), [dmtp_session_t](#) *session, const char *signame, const char *fingerprint, char **newprint)
- [PUBLIC_FUNC_DECL](#) (char *, [sgnt_resolv_dmtp_history](#), [dmtp_session_t](#) *session, const char *signame, const char *startfp, const char *endfp)
- [PUBLIC_FUNC_DECL](#) (char *, [sgnt_resolv_dmtp_stats](#), [dmtp_session_t](#) *session, const unsigned char *secret)
- [PUBLIC_FUNC_DECL](#) ([dmtp_mode_t](#), [sgnt_resolv_dmtp_str_to_mode](#), const char *modestr)
- [PUBLIC_FUNC_DECL](#) ([dmtp_mode_t](#), [sgnt_resolv_dmtp_get_mode](#), [dmtp_session_t](#) *session)

- [PUBLIC_FUNC_DECL](#) (int, sgnt_resolv_dmtp_noop, [dmtp_session_t](#) *session)
- [PUBLIC_FUNC_DECL](#) (int, sgnt_resolv_dmtp_reset, [dmtp_session_t](#) *session)
- [PUBLIC_FUNC_DECL](#) (char *, sgnt_resolv_dmtp_help, [dmtp_session_t](#) *session)
- [PUBLIC_FUNC_DECL](#) (int, sgnt_resolv_dmtp_quit, [dmtp_session_t](#) *session, int do_close)
- char * [_sgnt_resolv_read_dmtp_line](#) ([dmtp_session_t](#) *session, int *overflow, unsigned short *rcode, int *multiline)
Read a CR/LF-terminated line of input from an active DMTP session.
- char * [_sgnt_resolv_read_dmtp_multiline](#) ([dmtp_session_t](#) *session, int *overflow, unsigned short *rcode)
Read a multiple-line response from the DMTP server.
- char * [_sgnt_resolv_parse_line_code](#) (const char *line, unsigned short *rcode, int *multiline)
Parse a DMTP line into a numerical code and the trailing text.
- [dmtp_mode_t](#) [_sgnt_resolv_dmtp_str_to_mode](#) (const char *modestr)
Return the DMTP mode corresponding to a DMTP mode string.
- [dmtp_mode_t](#) [_sgnt_resolv_dmtp_initiate_starttls](#) ([dmtp_session_t](#) *session, const char *dxname)
Initiate a DMTP session over a plain TCP connection with the DMTP STARTTLS command.
- int [_sgnt_resolv_dmtp_expect_banner](#) ([dmtp_session_t](#) *session)
Attempt to receive a valid DMTP banner immediately after a connection is established.
- int [_sgnt_resolv_dmtp_issue_command](#) ([dmtp_session_t](#) *session, const char *cmd)
Issue a command to the remote server in a DMTP session.
- char * [_sgnt_resolv_dmtp_send_and_read](#) ([dmtp_session_t](#) *session, const char *cmd, unsigned short *rcode)
Issue a command to a DMTP server and get the response.
- int [_sgnt_resolv_dmtp_write_data](#) ([dmtp_session_t](#) *session, const void *buf, size_t buflen)
Write raw data to the remote end of a DMTP session.

5.146.1 Define Documentation

5.146.1.1 #define DMTP_LINE_BUF_SIZE 4096

Definition at line 17 of file dmtp.h.

5.146.1.2 #define DMTP_MAX_MX_RETRIES 3

Definition at line 15 of file dmtp.h.

Referenced by [_sgnt_resolv_dmtp_connect\(\)](#).

5.146.1.3 #define DMTP_PORT 26

Definition at line 10 of file dmtp.h.

Referenced by [_dx_connect_standard\(\)](#), and [_sgnt_resolv_dmtp_connect\(\)](#).

5.146.1.4 #define DMTP_PORT_DUAL 25

Definition at line 11 of file dmtp.h.

Referenced by [_dx_connect_dual\(\)](#), and [_sgnt_resolv_dmtp_connect\(\)](#).

5.146.1.5 `#define DMTP_V1_CIPHER_LIST "ECDHE-RSA-AES256-GCM-SHA384"`

Definition at line 13 of file `dmtp.h`.

Referenced by `_ssl_get_client_context()`.

5.146.2 Enumeration Type Documentation

5.146.2.1 `enum dmtp_mail_datatype_t`

Enumerator:

data_type_default

data_type_7bit

data_type_8bit

Definition at line 50 of file `dmtp.h`.

5.146.2.2 `enum dmtp_mail_rettype_t`

Enumerator:

return_type_default

return_type_full

return_type_display

return_type_header

Definition at line 43 of file `dmtp.h`.

5.146.2.3 `enum dmtp_mode_t`

Enumerator:

DMTP_MODE_DMTP

DMTP_MODE_SMTP

DMTP_MODE_NONE

dmtp_mode_unknown

dmtp_mode_dual

dmtp_mode_dmtp

dmtp_mode_smtp

dmtp_mode_esmtp

Definition at line 20 of file `dmtp.h`.

5.146.3 Function Documentation

5.146.3.1 `int _sgnt_resolv_dmtp_expect_banner (dmtp_session_t * session)`

Attempt to receive a valid DMTP banner immediately after a connection is established.

Parameters:

session a pointer to the DMTP session to have the server banner read.

Returns:

-1 on failure or 0 if a banner was successfully received advertising DMTPv1 compatibility.

Definition at line 1667 of file dmtp.c.

References `_dbgprint()`, `_sgnt_resolv_read_dmtp_line()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, and `RET_ERROR_INT`.

Referenced by `_dx_connect_dual()`, and `_dx_connect_standard()`.

5.146.3.2 dmtp_mode_t _sgnt_resolv_dmtp_initiate_starttls (dmtp_session_t * session, const char * dxname)

Initiate a DMTP session over a plain TCP connection with the DMTP STARTTLS command.

Note:

The `dxname` argument will be used as a parameter to the STARTTLS command for requesting a TLS certificate explicitly by hostname.

Parameters:

session a pointer to the DMTP session to have its transport security upgraded with TLS.

dxname the name of the DX server providing the TLS service of this DMTP session.

Returns:

`dmtp_mode_unknown` on failure or the active mode of the DMTP session as reported by the server on success.

Definition at line 1585 of file dmtp.c.

References `dmtp_session_t::fd`, `_sgnt_resolv_dmtp_str_to_mode()`, `_sgnt_resolv_read_dmtp_line()`, `_sgnt_resolv_read_dmtp_multiline()`, `_ssl_starttls()`, `chr_isspace`, `dmtp_session_t::con`, `dmtp_mode_dmtp`, `dmtp_mode_unknown`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `dmtp_session_t::mode`, `PUSH_ERROR_FMT`, and `RET_ERROR_CUST`.

Referenced by `_dx_connect_dual()`.

5.146.3.3 int _sgnt_resolv_dmtp_issue_command (dmtp_session_t * session, const char * cmd)

Issue a command to the remote server in a DMTP session.

Parameters:

session a pointer to the DMTP session across which the command will be issued.

cmd a null-terminated string containing the value of the specified DMTP command.

Returns:

-1 on failure or 0 if the command was successfully sent.

Definition at line 1715 of file dmtp.c.

References `_dbgprint()`, `dmtp_session_t::fd`, `dmtp_session_t::con`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, and `SSL_write_d`.

Referenced by `_sgnt_resolv_dmtp_ehlo()`, `_sgnt_resolv_dmtp_help()`, `_sgnt_resolv_dmtp_history()`, `_sgnt_resolv_dmtp_send_and_read()`, and `_sgnt_resolv_dmtp_stats()`.

5.146.3.4 char* _sgnt_resolv_dmtp_send_and_read (dmtp_session_t * session, const char * cmd, unsigned short * rcode)

Issue a command to a DMTP server and get the response.

Parameters:

session a pointer to the DMTP session across which the command will be issued.

cmd a null-terminated string containing the value of the specified DMTP command.

rcode

Returns:

NULL on failure or a pointer to a string containing the next line(s) of output from the DMTP server on success.

Definition at line 1756 of file dmtplib.c.

References `_sgnt_resolv_dmtplib_issue_command()`, `_sgnt_resolv_dmtplib_read_dmtplib_line()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, and `RET_ERROR_PTR`.

Referenced by `_sgnt_resolv_dmtplib_data()`, `_sgnt_resolv_dmtplib_get_mode()`, `_sgnt_resolv_dmtplib_get_signet()`, `_sgnt_resolv_dmtplib_mail_from()`, `_sgnt_resolv_dmtplib_noop()`, `_sgnt_resolv_dmtplib_quit()`, `_sgnt_resolv_dmtplib_rcpt_to()`, `_sgnt_resolv_dmtplib_reset()`, and `_sgnt_resolv_dmtplib_verify_signet()`.

5.146.3.5 dmtplib_mode_t _sgnt_resolv_dmtplib_str_to_mode (const char * modestr)

Return the DMTP mode corresponding to a DMTP mode string.

Parameters:

modestr a null-terminated string containing the human-readable name of the mode.

Returns:

the DMTP mode type corresponding to the supplied mode string, or `dmtplib_mode_unknown` otherwise.

Definition at line 1083 of file dmtplib.c.

References `dmtplib_mode_dmtplib`, `dmtplib_mode_esmtplib`, `dmtplib_mode_smtp`, and `dmtplib_mode_unknown`.

Referenced by `_sgnt_resolv_dmtplib_get_mode()`, and `_sgnt_resolv_dmtplib_initiate_starttls()`.

5.146.3.6 int _sgnt_resolv_dmtplib_write_data (dmtplib_session_t * session, const void * buf, size_t buflen)

Write raw data to the remote end of a DMTP session.

Parameters:

session a pointer to the DMTP session to which the data will be written.

buf a pointer to the buffer containing the raw data to be written.

buflen the size, in bytes, of the data buffer to be written to the DMTP session.

Returns:

0 if all requested bytes were written successfully to the connection, or -1 on failure.

Definition at line 1783 of file dmtplib.c.

References `_dbgprint()`, `dmtplib_session_t::fd`, `dmtplib_session_t::con`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, and `SSL_write_d`.

Referenced by `_sgnt_resolv_dmtplib_data()`.

5.146.3.7 `char* _sgnt_resolv_parse_line_code (const char * line, unsigned short * rcode, int * multiline)`

Parse a DMTP line into a numerical code and the trailing text.

Parameters:

line a pointer to a null-terminated string to be parsed as a DMTP response line or lines.

rcode an opointer to a variable that will receive the numeric response code of the DMTP reply.

multiline an optional parameter that if set will permit multiline DMTP responses. If this was the final line of a multiline response, the value will be set to 1 when the function returns, or 0 if there is more content to follow.

Returns:

NULL on failure or a pointer to a string containing the next line(s) of output from the DMTP server.

Definition at line 1524 of file dmtp.c.

References `chr_isspace`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `RET_ERROR_PTR`, and `RET_ERROR_PTR_FMT`.

Referenced by `_sgnt_resolv_read_dmtp_line()`.

5.146.3.8 `char* _sgnt_resolv_read_dmtp_line (dmtp_session_t * session, int * overflow, unsigned short * rcode, int * multiline)`

Read a CR/LF-terminated line of input from an active DMTP session.

Parameters:

session a pointer to the DMTP session from which the response line will be read.

overflow an optional pointer to a variable that will be set if the read operation exceeds the size of the internal line buffer.

rcode an optional pointer to a variable that will receive the numeric response code of the DMTP reply that was just received.

multiline an optional parameter that if set will permit multiline DMTP responses. If this was the final line of a multiline response, the value will be set to 1 when the function returns, or 0 if there is more content to follow.

Returns:

NULL on failure or a pointer to a string containing the next line(s) of output from the DMTP server.

Definition at line 1292 of file dmtp.c.

References `_dbgprint()`, `dmtp_session_t::fd`, `dmtp_session_t::inbuf`, `dmtp_session_t::inpos`, `_sgnt_resolv_parse_line_code()`, `_ssl_disconnect()`, `dmtp_session_t::active`, `dmtp_session_t::con`, `dmtp_mode_dmtp`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dmtp_session_t::mode`, `PUSH_ERROR_OPENSSL`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, and `SSL_read_d`.

Referenced by `_sgnt_resolv_dmtp_data()`, `_sgnt_resolv_dmtp_expect_banner()`, `_sgnt_resolv_dmtp_initiate_starttls()`, `_sgnt_resolv_dmtp_send_and_read()`, and `_sgnt_resolv_read_dmtp_multiline()`.

5.146.3.9 `char* _sgnt_resolv_read_dmtp_multiline (dmtp_session_t * session, int * overflow, unsigned short * rcode)`

Read a multiple-line response from the DMTP server.

Note:

A multiline response is designated by the presence of a hyphen between the response code and the responses text. A regular response, or the final line of a multiline response will instead have a space.

Parameters:

session a pointer to the DMTP session from which the response line(s) will be read.

overflow an optional pointer to a variable that will be set if the read operation exceeds the size of the internal line buffer.

rcode an optional pointer to a variable that will receive the numeric response code of the DMTP reply that was just received.

Returns:

NULL on failure or a pointer to a string containing the next response from the DMTP server on success.

Definition at line 1443 of file dmtplib.c.

References `_sgnt_resolv_read_dmtplib_line()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_sgnt_resolv_dmtplib_ehlo()`, `_sgnt_resolv_dmtplib_help()`, `_sgnt_resolv_dmtplib_history()`, `_sgnt_resolv_dmtplib_initiate_starttls()`, and `_sgnt_resolv_dmtplib_stats()`.

- 5.146.3.10 PUBLIC_FUNC_DECL (int, sgnt_resolv_dmtp_quit, dmtp_session_t * session, int do_close)
- 5.146.3.11 PUBLIC_FUNC_DECL (char *, sgnt_resolv_dmtp_help, dmtp_session_t * session)
- 5.146.3.12 PUBLIC_FUNC_DECL (int, sgnt_resolv_dmtp_reset, dmtp_session_t * session)
- 5.146.3.13 PUBLIC_FUNC_DECL (int, sgnt_resolv_dmtp_noop, dmtp_session_t * session)
- 5.146.3.14 PUBLIC_FUNC_DECL (dmtp_mode_t, sgnt_resolv_dmtp_get_mode, dmtp_session_t * session)
- 5.146.3.15 PUBLIC_FUNC_DECL (dmtp_mode_t, sgnt_resolv_dmtp_str_to_mode, const char * modestr)
- 5.146.3.16 PUBLIC_FUNC_DECL (char *, sgnt_resolv_dmtp_stats, dmtp_session_t * session, const unsigned char * secret)
- 5.146.3.17 PUBLIC_FUNC_DECL (char *, sgnt_resolv_dmtp_history, dmtp_session_t * session, const char * signame, const char * startfp, const char * endfp)
- 5.146.3.18 PUBLIC_FUNC_DECL (int, sgnt_resolv_dmtp_verify_signet, dmtp_session_t * session, const char * signame, const char * fingerprint, char ** newprint)
- 5.146.3.19 PUBLIC_FUNC_DECL (char *, sgnt_resolv_dmtp_get_signet, dmtp_session_t * session, const char * signame, const char * fingerprint)
- 5.146.3.20 PUBLIC_FUNC_DECL (char *, sgnt_resolv_dmtp_data, dmtp_session_t * session, void * msg, size_t msglen)
- 5.146.3.21 PUBLIC_FUNC_DECL (int, sgnt_resolv_dmtp_rcpt_to, dmtp_session_t * session, const char * domain)
- 5.146.3.22 PUBLIC_FUNC_DECL (int, sgnt_resolv_dmtp_mail_from, dmtp_session_t * session, const char * origin, size_t msgsize, dmtp_mail_rettype_t rettype, dmtp_mail_datatype_t dtype)
- 5.146.3.23 PUBLIC_FUNC_DECL (int, sgnt_resolv_dmtp_ehlo, dmtp_session_t * session, const char * domain)
- 5.146.3.24 PUBLIC_FUNC_DECL (int, verify_dx_certificate, dmtp_session_t * session)
- 5.146.3.25 PUBLIC_FUNC_DECL (dmtp_session_t *, dx_connect_dual, const char * host, const char * domain, int force_family, dime_record_t * dimerec, int failover)
- 5.146.3.26 PUBLIC_FUNC_DECL (dmtp_session_t *, dx_connect_standard, const char * host, const char * domain, int force_family, dime_record_t * dimerec)
- 5.146.3.27 PUBLIC_FUNC_DECL (void, sgnt_resolv_destroy_dmtp_session, dmtp_session_t * session)
- 5.146.3.28 PUBLIC_FUNC_DECL (dmtp_session_t *, sgnt_resolv_dmtp_connect, const char * domain, int force_family)
- 5.146.3.29 PUBLIC_FUNC_DECL (signet_t *, get_signet, const char * name, const char * fingerprint, int use_cache)

5.147 src/servers/dmtp/dmtp.h File Reference

Functions

- void [dmtp_requeue](#) ([connection_t](#) *con)
commands.c
- [int_t dmtp_compare](#) (const void *compare, const void *command)
- void [dmtp_process](#) ([connection_t](#) *con)
The main entry point in the DMTP server for processing commands issued by clients.
- void [dmtp_sort](#) (void)
Sort the DMTP command table to be ready for binary searches.
- void [dmtp_ehlo](#) ([connection_t](#) *con)
dmtp.c
- void [dmtp_helo](#) ([connection_t](#) *con)
Process an DMTP HELO command.
- void [dmtp_noop](#) ([connection_t](#) *con)
Perform an DMTP NOOP (no-operation) command.
- void [dmtp_mode](#) ([connection_t](#) *con)
- void [dmtp_rset](#) ([connection_t](#) *con)
Reset the DMTP session, in response to an DMTP RSET command.
- void [dmtp_quit](#) ([connection_t](#) *con)
Gracefully terminate an DMTP session, especially in response to an DMTP QUIT command.
- void [dmtp_mail](#) ([connection_t](#) *con)
Specify the origin domain for a message in response to an DMTP MAIL command.
- void [dmtp_rcpt](#) ([connection_t](#) *con)
Specify the destination domain for a message in response to an DMTP RCPT command.
- void [dmtp_data](#) ([connection_t](#) *con)
Process an DMTP MAIL command.
- void [dmtp_sgnt](#) ([connection_t](#) *con)
- void [dmtp_hist](#) ([connection_t](#) *con)
- void [dmtp_vrfy](#) ([connection_t](#) *con)
- void [dmtp_help](#) ([connection_t](#) *con)
- void [dmtp_verb](#) ([connection_t](#) *con)
- void [dmtp_init](#) ([connection_t](#) *con)
The start of the protocol handler for the DMTP server.
- void [dmtp_invalid](#) ([connection_t](#) *con)
A function that is executed when an invalid DMTP command is executed.
- void [dmtp_session_reset](#) ([connection_t](#) *con)
session.c

- void [dmtp_session_destroy](#) ([connection_t](#) *con)
Destroy the data associated with an DMTP session.

5.147.1 Function Documentation

5.147.1.1 [int_t dmtp_compare](#) (const void * *compare*, const void * *command*)

Definition at line 11 of file `commands.c`.

References `command_t`, `PLACER`, `st_cmp_ci_eq()`, and `st_cmp_ci_starts()`.

Referenced by `dmtp_process()`, and `dmtp_sort()`.

5.147.1.2 [void dmtp_data](#) ([connection_t](#) * *con*)

Process an DMTP MAIL command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 123 of file `dmtp.c`.

References `con_write_bl()`, and `dmtp_requeue()`.

Referenced by `dmtp_process()`.

5.147.1.3 [void dmtp_ehlo](#) ([connection_t](#) * *con*)

`dmtp.c dmtp.c`

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 15 of file `dmtp.c`.

References `con_print()`, `st_char_get()`, and `st_length_int()`.

5.147.1.4 [void dmtp_helo](#) ([connection_t](#) * *con*)

Process an DMTP HELO command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 28 of file `dmtp.c`.

References `con_print()`, `st_char_get()`, and `st_length_int()`.

5.147.1.5 void dmtp_help (connection_t * con)

Definition at line 158 of file dmtp.c.

References con_write_bl().

5.147.1.6 void dmtp_hist (connection_t * con)

Definition at line 144 of file dmtp.c.

References con_write_bl().

5.147.1.7 void dmtp_init (connection_t * con)

The start of the protocol handler for the DMTP server.

Parameters:

con the new inbound DMTP client connection.

Returns:

This function returns no value.

Definition at line 175 of file dmtp.c.

References con_print(), con_reverse_enqueue(), dmtp_requeue(), st_char_get(), and st_length_int().

Referenced by protocol_enqueue().

5.147.1.8 void dmtp_invalid (connection_t * con)

A function that is executed when an invalid DMTP command is executed.

Returns:

This function returns no value.

Definition at line 52 of file dmtp.c.

References con_write_bl().

Referenced by dmtp_process().

5.147.1.9 void dmtp_mail (connection_t * con)

Specify the origin domain for a message in response to an DMTP MAIL command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 111 of file dmtp.c.

References con_write_bl().

5.147.1.10 void dmtp_mode (connection_t * con)

Definition at line 130 of file dmtp.c.

References `con_write_bl()`.

5.147.1.11 void dmtp_noop (connection_t * con)

Perform an DMTP NOOP (no-operation) command.

Note:

This command does essentially nothing and is mostly a way to keep connections alive without timing out due to inactivity.

Returns:

This function returns no value.

Definition at line 42 of file dmtp.c.

References `con_write_bl()`.

5.147.1.12 void dmtp_process (connection_t * con)

The main entry point in the DMTP server for processing commands issued by clients.

Parameters:

con a pointer to the connection object of the client issuing the DMTP command.

Returns:

This function returns no value.

Definition at line 48 of file commands.c.

References `command`, `command_t`, `con_read_line()`, `dmtp_commands`, `dmtp_compare()`, `dmtp_data()`, `dmtp_invalid()`, `dmtp_process()`, `dmtp_quit()`, `dmtp_requeue()`, `enqueue()`, `pl_char_get()`, `pl_empty()`, `pl_length_get()`, and `requeue()`.

Referenced by `dmtp_process()`, and `dmtp_requeue()`.

5.147.1.13 void dmtp_quit (connection_t * con)

Gracefully terminate an DMTP session, especially in response to an DMTP QUIT command.

Parameters:

the DMTP client connection to be terminated.

Returns:

This function returns no value.

Definition at line 65 of file dmtp.c.

References `con_destroy()`, `con_status()`, and `con_write_bl()`.

Referenced by `dmtp_process()`, and `dmtp_requeue()`.

5.147.1.14 void dmtplib_rcpt (connection_t * con)

Specify the destination domain for a message in response to an DMTP RCPT command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 100 of file dmtplib.c.

References con_write_bl().

5.147.1.15 void dmtplib_requeue (connection_t * con)

commands.c

Definition at line 31 of file commands.c.

References con_status(), dmtplib_process(), dmtplib_quit(), enqueue(), and status.

Referenced by dmtplib_data(), dmtplib_init(), and dmtplib_process().

5.147.1.16 void dmtplib_rset (connection_t * con)

Reset the DMTP session, in response to an DMTP RSET command.

Note:

This command clears any sender, recipient, and mail data, along with all buffers and state tables. The connection structure is reset to the same it was in immediately after the HELO/EHLO command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 88 of file dmtplib.c.

References con_write_bl(), and dmtplib_session_reset().

5.147.1.17 void dmtplib_session_destroy (connection_t * con)

Destroy the data associated with an DMTP session.

Parameters:

con the DMTP client connection to be destroyed.

Returns:

This function returns no value.

Definition at line 25 of file session.c.

Referenced by con_destroy().

5.147.1.18 void dmtp_session_reset (connection_t * con)

session.c session.c

Parameters:

con the DMTP client connection to be reset.

Returns:

This function returns no value.

Definition at line 15 of file session.c.

Referenced by dmtp_rset().

5.147.1.19 void dmtp_sgnt (connection_t * con)

Definition at line 137 of file dmtp.c.

References con_write_bl().

5.147.1.20 void dmtp_sort (void)

Sort the DMTP command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 26 of file commands.c.

References command_t, dmtp_commands, and dmtp_compare().

Referenced by protocol_init().

5.147.1.21 void dmtp_verb (connection_t * con)

Definition at line 164 of file dmtp.c.

References con_write_bl().

5.147.1.22 void dmtp_vrfy (connection_t * con)

Definition at line 151 of file dmtp.c.

References con_write_bl().

5.148 src/network/http.h File Reference

Data Structures

- struct [http_data_t](#)
- struct [http_content_t](#)
- struct [http_page_t](#)
- struct [__attribute__](#)

Enumerations

- enum [http_method_t](#) {
 [HTTP_METHOD_NONE](#), [HTTP_METHOD_GET](#), [HTTP_METHOD_POST](#), [HTTP_METHOD_PUT](#),
 [HTTP_METHOD_DELETE](#), [HTTP_METHOD_HEAD](#), [HTTP_METHOD_TRACE](#), [HTTP_METHOD_OPTIONS](#),
 [HTTP_METHOD_CONNECT](#), [HTTP_METHOD_UNSUPPORTED](#) }
- enum [HTTP_DATA](#) { [HTTP_DATA_ANY](#), [HTTP_DATA_HEADER](#), [HTTP_DATA_GET](#), [HTTP_DATA_POST](#) }
- enum { [HTTP_MERGED](#), [HTTP_PORTAL](#) }

5.148.1 Enumeration Type Documentation

5.148.1.1 anonymous enum

Enumerator:

[HTTP_MERGED](#)
[HTTP_PORTAL](#)

Definition at line 31 of file `http.h`.

5.148.1.2 enum `HTTP_DATA`

Enumerator:

[HTTP_DATA_ANY](#)
[HTTP_DATA_HEADER](#)
[HTTP_DATA_GET](#)
[HTTP_DATA_POST](#)

Definition at line 24 of file `http.h`.

5.148.1.3 enum `http_method_t`

Enumerator:

[HTTP_METHOD_NONE](#)
[HTTP_METHOD_GET](#)
[HTTP_METHOD_POST](#)
[HTTP_METHOD_PUT](#)
[HTTP_METHOD_DELETE](#)
[HTTP_METHOD_HEAD](#)
[HTTP_METHOD_TRACE](#)

HTTP_METHOD_OPTIONS

HTTP_METHOD_CONNECT

HTTP_METHOD_UNSUPPORTED

Definition at line 11 of file http.h.

5.149 src/servers/http/http.h File Reference

Enumerations

- enum { [HTTP_CONNECTION_CLOSE](#) = -1, [HTTP_CONNECTION_NEUTRAL](#) = 0, [HTTP_CONNECTION_KEEPALIVE](#) = 1 }
- enum { [HTTP_COOKIE_DELETE](#) = -1, [HTTP_COOKIE_NEUTRAL](#) = 0, [HTTP_COOKIE_SET](#) = 1 }
- enum {
[HTTP_READY](#) = 0, [HTTP_PARSE_HEADER](#) = 1, [HTTP_PARSE_PAIRS](#) = 2, [HTTP_PARSE_COOKIE](#) = 3,
[HTTP_READ_BODY](#) = 8, [HTTP_RESPOND](#) = 10, [HTTP_COMPLETE](#) = 12, [HTTP_OK](#) = 200,
[HTTP_ERROR_400](#) = 400, [HTTP_ERROR_401](#) = 401, [HTTP_ERROR_403](#) = 403, [HTTP_ERROR_404](#) = 404,
[HTTP_ERROR_405](#) = 405, [HTTP_ERROR_422](#) = 422, [HTTP_ERROR_500](#) = 500, [HTTP_ERROR_501](#) = 501,
[HTTP_CLOSE](#) = 1000 }

Functions

- [bool_t http_content_load_directory](#) ([int_t](#) template, [chr_t](#) *directory)
[content.c](#)
- [bool_t http_content_load_fonts](#) (void)
Load all fonts into the http server repository.
- [bool_t http_content_refresh](#) (void)
Refresh the stored contents of the web app templates, static content, and fonts directories.
- [bool_t http_content_start](#) (void)
Prime the the web app templates, static content, and fonts directories for future use.
- void [http_content_stop](#) (void)
Purge the http contents repository of stored fonts, templates, and static web pages.
- void [http_free_content](#) ([http_content_t](#) *page)
LOW: We should use basename() and dirname() to cleanup path strings.
- [http_content_t](#) * [http_get_static](#) ([stringer_t](#) *location)
Get a cached copy of a static web page.
- [http_content_t](#) * [http_get_template](#) ([chr_t](#) *location)
Return a template by location.
- [bool_t http_load_file](#) ([int_t](#) template, [chr_t](#) *filename)
Load file content into the http server repository.
- void [http_page_free](#) ([http_page_t](#) *page)
Free an http page object and all its underlying data.
- [http_page_t](#) * [http_page_get](#) ([chr_t](#) *location)
Get a template page and prepare its xml document root for use.
- void [http_data_free](#) ([http_data_t](#) *data)
[data.c](#)

- [http_data_t * http_data_get](#) ([connection_t *con](#), [HTTP_DATA](#) source, [chr_t *name](#))
Get a name/value pair associated with an http connection, by name.
- [http_data_t * http_data_header_parse_line](#) ([chr_t *buf](#), [size_t len](#))
Parse a data buffer into a http header name/value pair.
- [http_data_t * http_data_header_parse](#) ([connection_t *con](#))
Parse the current line of input from an http client connection into a http header name/value pair.
- [void http_data_value_decode](#) ([stringer_t *string](#))
Decode an escaped URI component into its original data.
- [int_t http_data_value_parse](#) ([connection_t *con](#), [HTTP_DATA](#) source, [placer_t](#) pair)
Parse a string containing a name/value pair; and place it in the specified connection's pairs holder.
- [void http_print_301](#) ([connection_t *con](#), [chr_t *location](#), [int_t](#) tls)
errors.c
- [void http_print_400](#) ([connection_t *con](#))
Return an HTTP 400 bad client request response to the client.
- [void http_print_403](#) ([connection_t *con](#))
Return an HTTP 403 access denied response to the client.
- [void http_print_404](#) ([connection_t *con](#))
Return an HTTP 404 location not found response to the client.
- [void http_print_405](#) ([connection_t *con](#))
Return an HTTP 405 method not allowed response to the client.
- [void http_print_500](#) ([connection_t *con](#))
Return an HTTP 500 internal server error response to the client.
- [void http_print_500_log](#) ([connection_t *con](#), [chr_t *logmsg](#))
Return an HTTP 500 internal server error response to the client, with additional logging information.
- [void http_print_501](#) ([connection_t *con](#))
Return an HTTP 501 method not implemented response to the client.
- [void http_body](#) ([connection_t *con](#))
http.c
- [void http_close](#) ([connection_t *con](#))
Close a connection corresponding to an http session.
- [void http_init](#) ([connection_t *con](#))
Handle a new http client connection.
- [void http_process](#) ([connection_t *con](#))
Process data sent by an http client.
- [void http_requeue](#) ([connection_t *con](#))
The main http server requeue entry point state machine for processing client data.

- void `http_parse_context` (`connection_t` *con, `stringer_t` *application, `stringer_t` *path)
Attempt to retrieve a connected user's associated session, by searching the cookie, the POST "session" variable, and the URL.
- void `http_parse_header` (`connection_t` *con)
Parse and process the current line of input from an http client connection as an http request header, storing data in the connection's `http.headers` member.
- void `http_parse_method` (`connection_t` *con)
Parse an http request and determine the request method and location.
- `int_t` `http_parse_origin` (`stringer_t` *s, `placer_t` *output)
Get the origin of a resource from a url.
- void `http_parse_pairs` (`connection_t` *con)
Parse all the GET and POST parameters present in an http client request, and store them with the connection.
- `placer_t` `get_header_value_noopt` (`stringer_t` *vstring)
Get the simple value of an http header, with any optional parameters stripped away.
- `placer_t` `get_header_opt` (`stringer_t` *vstring, `stringer_t` *optname)
Get the value of a named optional parameter from an http header value.
- `bool_t` `multipart_get_boundary` (`connection_t` *con, `placer_t` *output)
Get the boundary delimiter for a request by a connection specifying a Content-Type of multipart/form-data.
- void `http_response` (`connection_t` *con)
response.c
- `stringer_t` * `http_response_allow_cross` (`connection_t` *con)
- `stringer_t` * `http_response_connection` (`connection_t` *con, `int_t` force)
Get an appropriate value for the Connection header of an http response.
- `stringer_t` * `http_response_cookie` (`connection_t` *con)
- void `http_response_header` (`connection_t` *con, `int_t` status, `stringer_t` *type, `size_t` len)
Send a full set of http response headers to the remote client.
- void `http_response_options` (`connection_t` *con)
Return a response to an http OPTIONS request.
- `chr_t` * `http_response_status` (`int_t` status)
Get a descriptive string for a numerical http status code.
- void `http_session_destroy` (`connection_t` *con)
sessions.c
- void `http_session_reset` (`connection_t` *con)
Reset a client http connection to its original, uninitialized state.

5.149.1 Enumeration Type Documentation

5.149.1.1 anonymous enum

Enumerator:

HTTP_CONNECTION_CLOSE
HTTP_CONNECTION_NEUTRAL
HTTP_CONNECTION_KEEPALIVE

Definition at line 11 of file http.h.

5.149.1.2 anonymous enum

Enumerator:

HTTP_COOKIE_DELETE
HTTP_COOKIE_NEUTRAL
HTTP_COOKIE_SET

Definition at line 17 of file http.h.

5.149.1.3 anonymous enum

Enumerator:

HTTP_READY
HTTP_PARSE_HEADER
HTTP_PARSE_PAIRS
HTTP_PARSE_COOKIE
HTTP_READ_BODY
HTTP_RESPOND
HTTP_COMPLETE
HTTP_OK
HTTP_ERROR_400
HTTP_ERROR_401
HTTP_ERROR_403
HTTP_ERROR_404
HTTP_ERROR_405
HTTP_ERROR_422
HTTP_ERROR_500
HTTP_ERROR_501
HTTP_CLOSE

Definition at line 23 of file http.h.

5.149.2 Function Documentation

5.149.2.1 `placer_t get_header_opt (stringer_t * vstring, stringer_t * optname)`

Get the value of a named optional parameter from an http header value.

Note:

Only the value after the ":" in a header field is passed to this function (in other words, the header name is omitted).

Parameters:

vstring a managed string containing the single http header line value to be parsed.

optname a managed string with the name of the optional parameter to be found.

Returns:

a placer pointing to the named optional parameter of the specified http header value.

Definition at line 336 of file `parse.c`.

References `pl_null()`, `pl_trim_start()`, `placer_t`, `st_cmp_cs_eq()`, `tok_get_count_st()`, and `tok_get_st()`.

Referenced by `multipart_get_boundary()`, and `portal_upload()`.

5.149.2.2 `placer_t get_header_value_noopt (stringer_t * vstring)`

Get the simple value of an http header, with any optional parameters stripped away.

Note:

Only the value after the ":" in a header field is passed to this function (in other words, the header name is omitted).

Parameters:

vstring a managed string containing the single http header line value to be parsed.

Returns:

a placer pointing to the option-free header value of the specified input line.

Definition at line 313 of file `parse.c`.

References `pl_init()`, `placer_t`, `st_char_get()`, `st_length_get()`, `tok_get_count_st()`, and `tok_get_st()`.

Referenced by `http_parse_header()`, and `multipart_get_boundary()`.

5.149.2.3 `void http_body (connection_t * con)`

[http.c](#) `http.c`

Note:

Any request errors will be handled directly without returns. This function sets the value of the connection's `http.body` member.

Parameters:

con a pointer to the connection object of the remote http client.

Returns:

This function returns no value.

Definition at line 73 of file http.c.

References `con_read()`, `CONTIGUOUS`, `data`, `HEAP`, `magma_t::http`, `http_data_get()`, `HTTP_DATA_HEADER`, `HTTP_ERROR_400`, `HTTP_RESPOND`, `JOINTED`, `length`, `magma_t::log`, `log_pedantic`, `magma`, `MANAGED_T`, `MAPPED_T`, `size_conv_bl()`, `st_alloc_opts()`, `st_append_opts()`, `st_char_get()`, `st_cleanup`, `st_data_get()`, `st_dupe_opts()`, `st_empty`, `st_length_get()`, `st_length_int()`, and `http_data_t::value`.

Referenced by `http_requeue()`.

5.149.2.4 void http_close (connection_t * con)

Close a connection corresponding to an http session.

Returns:

This function returns no value.

Definition at line 14 of file http.c.

References `con_destroy()`.

Referenced by `http_process()`, and `http_requeue()`.

5.149.2.5 bool_t http_content_load_directory (int_t template, chr_t * directory)

[content.c](#) [content.c](#)

See also:

[http_load_file\(\)](#)

Parameters:

template a value specifying whether the specified directory contains templates or regular web content.

directory a null-terminated string specifying the pathname of the directory to be scanned recursively.

Returns:

0 on failure or 1 on success.

Definition at line 313 of file content.c.

References `http_content_load_directory()`, `http_load_file()`, `log_info`, `MEMORYBUF`, `NULLER`, `PLACER`, `st_char_get()`, `st_cmp_cs_ends()`, `st_cmp_cs_eq()`, `st_free()`, and `st_merge`.

Referenced by `http_content_load_directory()`, `http_content_refresh()`, and `http_content_start()`.

5.149.2.6 bool_t http_content_load_fonts (void)

Load all fonts into the http server repository.

Note:

This function will load all fonts of extension ".ttf" from the directory configured in [magma.http.fonts](#).

Returns:

0 on failure or 1 on success.

Definition at line 362 of file content.c.

References `magma_t::content`, `content`, `magma_t::http`, `inx_insert()`, `magma_t::log`, `log_info`, `log_pedantic`, `M_TYPE_UINT64`, `magma`, `MEMORYBUF`, `NULLER`, `PLACER`, `st_cmp_ci_ends()`, `st_cmp_cs_ends()`, `st_free()`, `st_merge`, `multi_t::u64`, and `multi_t::val`.

Referenced by `http_content_refresh()`, and `http_content_start()`.

5.149.2.7 bool_t http_content_refresh (void)

Refresh the stored contents of the web app templates, static content, and fonts directories.

Returns:

true if all content directories were successfully refreshed, or false on failure.

Definition at line 449 of file content.c.

References content, magma_t::http, http_content_load_directory(), http_content_load_fonts(), http_free_content(), inx_alloc(), inx_free(), M_INX_LINKED, magma, and st_free().

Referenced by signal_refresh().

5.149.2.8 bool_t http_content_start (void)

Prime the the web app templates, static content, and fonts directories for future use.

Note:

This function makes sure that the web templates, static content, and fonts directory have been properly set, and loads and caches their content for future use.

Returns:

true on success or false on failure.

Definition at line 407 of file content.c.

References content, magma_t::http, http_content_load_directory(), http_content_load_fonts(), http_free_content(), inx_alloc(), log_pedantic, M_INX_LINKED, magma, NULLER, PLACER, st_cmp_cs_ends(), and st_free().

Referenced by process_start().

5.149.2.9 void http_content_stop (void)

Purge the http contents repository of stored fonts, templates, and static web pages.

Returns:

This function returns no value.

Definition at line 435 of file content.c.

References content, and inx_cleanup().

Referenced by process_stop().

5.149.2.10 void http_data_free (http_data_t * data)

data.c data.c

Parameters:

data the http data object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file data.c.

References `mm_free()`, `http_data_t::name`, `st_cleanup`, and `http_data_t::value`.

Referenced by `http_data_value_parse()`, `http_parse_header()`, `http_parse_pairs()`, and `portal_upload()`.

5.149.2.11 `http_data_t* http_data_get (connection_t * con, HTTP_DATA source, chr_t * name)`

Get a name/value pair associated with an http connection, by name.

Note:

The name/value pairs searched can be supplied by a client through http request headers, or through GET or POST data.

Parameters:

con the connection object to be queried.

source the source of the client supplied value pair: `HTTP_DATA_GET`, `HTTP_DATA_POST` or `HTTP_DATA_ANY`.

name the name associated with the name/value pair to be retrieved.

Returns:

NULL on failure, or a pointer to the http name/value data pair requested on success.

TODO: We could just use the index find function.

Definition at line 34 of file data.c.

References `data`, `HTTP_DATA_ANY`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `http_data_t::name`, `NULLER`, `http_data_t::source`, and `st_cmp_ci_eq()`.

Referenced by `contact_business()`, `contact_print_form()`, `contact_process()`, `http_body()`, `http_response_allow_cross()`, `multipart_get_boundary()`, `register_business_step1()`, `register_business_step2()`, `register_process()`, and `teacher_process()`.

5.149.2.12 `http_data_t* http_data_header_parse (connection_t * con)`

Parse the current line of input from an http client connection into a http header name/value pair.

Parameters:

con the connection object of the http client to be read.

Returns:

NULL on failure, or a pointer to an http header name/value pair on success.

Definition at line 226 of file data.c.

References `http_data_header_parse_line()`, `st_char_get()`, and `st_length_get()`.

Referenced by `http_parse_header()`.

5.149.2.13 `http_data_t* http_data_header_parse_line (chr_t * buf, size_t len)`

Parse a data buffer into a http header name/value pair.

Parameters:

con the connection object of the http client to be read.

Returns:

NULL on failure, or a pointer to an http header name/value pair on success.

Definition at line 163 of file data.c.

References HTTP_DATA_HEADER, mm_alloc(), http_data_t::name, http_data_t::source, st_free(), st_import(), and http_data_t::value.

Referenced by http_data_header_parse(), and portal_upload().

5.149.2.14 void http_data_value_decode (stringer_t * *string*)

Decode an escaped URI component into its original data.

Note:

Since the un-escaped string will always be at least as small as the encoded value, the input managed string is transformed in place.

Parameters:

string a managed string containing the escaped URI data to be decoded.

Returns:

This function returns no value.

Definition at line 68 of file data.c.

References hex_decode_chr(), length, st_char_get(), st_length_get(), and st_length_set().

Referenced by http_data_value_parse().

5.149.2.15 int_t http_data_value_parse (connection_t * *con*, HTTP_DATA *source*, placer_t *pair*)

Parse a string containing a name/value pair, and place it in the specified connection's pairs holder.

Note:

Each name/value pair is assumed to be delimited with a '=' character, and each half is URI-decoded before storage.

Parameters:

con a pointer to the connection object of the http client submitting the user data to be parsed.

source an HTTP_DATA value specifying the source of the name/value pair (can be HTTP_DATA_HEADER, HTTP_DATA_GET, or HTTP_DATA_POST).

pair a placer pointing to a string containing the name/value pair to be parsed.

Returns:

0 on general or parsing failure, or 1 if the specified input buffer was successfully parsed and stored.

Definition at line 111 of file data.c.

References CONTIGUOUS, data, HEAP, magma_t::http, http_data_free(), http_data_value_decode(), inx_insert(), magma_t::log, log_pedantic, M_TYPE_STRINGER, magma, MANAGED_T, mm_alloc(), http_data_t::name, pl_empty(), placer_t, http_data_t::source, multi_t::st, st_char_get(), st_dupe_opts(), st_free(), st_length_int(), tok_get_pl(), multi_t::val, and http_data_t::value.

Referenced by http_parse_pairs().

5.149.2.16 void http_free_content (http_content_t * *page*)

LOW: We should use basename() and dirname() to cleanup path strings. LOW: These functions should all be renamed to http_content_XXXX. The file and directory functions should be updated to use the x64/reentrant alternatives. Specifically open64/fstat64/readdir64_r. An update function that can be triggered with a SIGHUP would also be nice. Free a stored piece of http content and all its underlying data.

Parameters:

page a pointer to the loaded content page to be freed.

Returns:

This function returns no value.

Definition at line 27 of file content.c.

References http_content_t::location, mm_free(), http_content_t::resource, st_cleanup, and http_content_t::type.

Referenced by http_content_refresh(), http_content_start(), and http_load_file().

5.149.2.17 http_content_t* http_get_static (stringer_t * *location*)

Get a cached copy of a static web page.

Parameters:

location a pointer to a managed string containing the location of the requested resource.

Returns:

NULL on failure, or a pointer to an http content object with the contents of the requested resource on success.

Definition at line 67 of file content.c.

References content, inx_find(), M_TYPE_STRINGER, and http_content_t::type.

Referenced by http_response().

5.149.2.18 http_content_t* http_get_template (chr_t * *location*)

Return a template by location.

Parameters:

location a string specifying the location of the template.

Returns:

NULL on failure, or a pointer to an [http_content_t](#) object containing the template.

Definition at line 83 of file content.c.

References content, inx_find(), M_TYPE_STRINGER, NULLER, and http_content_t::type.

Referenced by http_page_get(), register_print_message(), register_print_step1(), register_print_step2(), and register_print_step3().

5.149.2.19 void http_init (connection_t * *con*)

Handle a new http client connection.

Parameters:

con a pointer to the http client connection that was just accepted.

Returns:

This function returns no value.

Definition at line 178 of file http.c.

References `con_reverse_enqueue()`, and `http_process()`.

Referenced by `protocol_enqueue()`.

5.149.2.20 bool_t http_load_file (int_t template, chr_t *filename)

Load file content into the http server repository.

Note:

Each file that is loaded will be cached for retrieval by http clients, with its mime type determined automatically. Any files passed as a template will have the ".template" extension trimmed from the end of its resource path, and any file named 'index.html' will be read as the default directory index path. Each file will also be added to its respective parent page or template inx holder.

Parameters:

template a value specifying whether or not the specified file is a template.

filename a null-terminated string specifying the pathname of the file to be loaded.

Returns:

0 on failure or 1 on success.

Definition at line 149 of file content.c.

References `magma_t::content`, `content`, `CONTIGUOUS`, `data`, `HEAP`, `magma_t::http`, `http_free_content()`, `inx_insert()`, `magma_t::log`, `log_info`, `log_pedantic`, `M_TYPE_STRINGER`, `magma`, `MANAGED_T`, `MEMORYBUF`, `mm_alloc()`, `ns_length_get()`, `NULLER`, `PLACER`, `multi_t::st`, `st_alloc()`, `st_char_get()`, `st_cmp_ci_ends()`, `st_cmp_cs_ends()`, `st_cmp_cs_eq()`, `st_dupe()`, `st_dupe_opts()`, `st_free()`, `st_import()`, `st_length_get()`, `st_length_int()`, `st_length_set()`, and `multi_t::val`.

Referenced by `http_content_load_directory()`.

5.149.2.21 void http_page_free (http_page_t *page)

Free an http page object and all its underlying data.

Parameters:

page a pointer to the http page object to be freed.

Definition at line 43 of file content.c.

References `http_page_t::doc_ctx`, `http_page_t::doc_obj`, `mm_free()`, `xml_free_doc()`, `xml_free_parser_ctx()`, `xml_free_xpath_ctx()`, and `http_page_t::xpath_ctx`.

Referenced by `contact_print_form()`, `contact_print_message()`, `http_page_get()`, `portal_print_login()`, `statistics_process()`, `teacher_print_form()`, and `teacher_print_message()`.

5.149.2.22 http_page_t* http_page_get (chr_t * location)

Get a template page and prepare its xml document root for use.

Note:

Each page is affixed with an xpath with a namespace after passing through the xml parser.

Parameters:

location a pointer to a null-terminated string with the pathname of the template to be returned.

Returns:

NULL on failure, or a pointer to the http page object of the requested template.

Definition at line 100 of file content.c.

References `http_page_t::content`, `http_page_t::doc_ctx`, `http_page_t::doc_obj`, `http_get_template()`, `http_page_free()`, `log_pedantic`, `mm_alloc()`, `http_content_t::resource`, `st_char_get()`, `st_length_get()`, `xml_create_doc()`, `xml_create_parser_ctx()`, `xml_create_xpath_ctx()`, `xml_xpath_set_namespace()`, and `http_page_t::xpath_ctx`.

Referenced by `contact_business_add_error()`, `contact_print_form()`, `contact_print_message()`, `portal_print_login()`, `statistics_process()`, `teacher_add_error()`, `teacher_print_form()`, and `teacher_print_message()`.

5.149.2.23 void http_parse_context (connection_t * con, stringer_t * application, stringer_t * path)

Attempt to retrieve a connected user's associated session, by searching the cookie, the POST "session" variable, and the URL. TODO: The header and body parsers need limits on how many headers/bytes they will accept. `parse.c`

Parameters:

con a pointer to the connection object to be queried for a session id.

application a managed string containing the application associated with the connection's pending request.

path a managed string containing the path associated with the connection's pending request.

Returns:

This function returns no value.

TODO: Develop better logic for handling cookies. Namely add the ability to forcibly trigger the Set-Cookie entity and if necessary delete the existing cookie.

Definition at line 115 of file `parse.c`.

References `HTTP_METHOD_POST`, `json_decref_d`, `json_loads_d`, `json_object_get_d`, `json_string_value_d`, `log_pedantic`, `NULLER`, `pl_init()`, `placer_t`, `sess_get()`, `st_char_get()`, `st_cmp_ci_starts()`, `st_length_get()`, `tok_get_pl()`, and `tok_get_st()`.

Referenced by `json_api_dispatch()`, `portal_endpoint()`, `portal_process()`, and `portal_upload()`.

5.149.2.24 void http_parse_header (connection_t * con)

Parse and process the current line of input from an http client connection as an http request header, storing data in the connection's `http.headers` member.

Note:

If no more http headers can be read, control is returned by setting the connection `http.mode` to `HTTP_RESPOND`. Special actions are taken to store the "Host", "User-Agent", "Cookie", and "Connection" headers.

Parameters:

con the connection object of the http client to be read, and to store the results of the operation.

Returns:

This function returns no value.

LOW: Should we bother to throw an error if con->http.connection isn't HTTP_CONNECTION_NEUTRAL?

Definition at line 177 of file parse.c.

References con_print(), CONTIGUOUS, data, get_header_value_noopt(), HEAP, magma_t::http, HTTP_CONNECTION_CLOSE, HTTP_CONNECTION_KEEPALIVE, http_data_free(), http_data_header_parse(), HTTP_ERROR_500, HTTP_RESPOND, int32_conv_bl(), inx_alloc(), inx_insert(), magma_t::log, log_info, lower_st(), M_INX_LINKED, M_TYPE_STRINGER, magma, MANAGED_T, http_data_t::name, PLACER, placer_t, multi_t::st, st_char_get(), st_cmp_ci_eq(), st_dupe_opts(), st_length_get(), st_length_int(), st_length_set(), st_search_cs(), multi_t::val, and http_data_t::value.

Referenced by http_process().

5.149.2.25 void http_parse_method (connection_t * con)

Parse an http request and determine the request method and location.

Note:

This function returns no value but sets the internal method, location, and state of the underlying connection object.

Parameters:

con the client connection making an http request.

Returns:

This function returns no value.

Definition at line 269 of file parse.c.

References CONTIGUOUS, HEAP, magma_t::http, HTTP_METHOD_CONNECT, HTTP_METHOD_DELETE, HTTP_METHOD_GET, HTTP_METHOD_HEAD, HTTP_METHOD_OPTIONS, HTTP_METHOD_POST, HTTP_METHOD_PUT, HTTP_METHOD_TRACE, HTTP_METHOD_UNSUPPORTED, HTTP_PARSE_HEADER, magma_t::log, log_info, magma, MANAGED_T, PLACER, placer_t, st_char_get(), st_cmp_ci_starts(), st_dupe_opts(), st_length_int(), tok_get_count_st(), and tok_get_pl().

Referenced by http_process().

5.149.2.26 int_t http_parse_origin (stringer_t * s, placer_t * output)

Get the origin of a resource from a url.

Note:

Usually we'll be give the value of the Origin header field to work with so we'll end up returning the entire string, but if we had to fall back and use the Referrer instead this should strip off the path portion.

Parameters:

s a managed string containing the url to be parsed.

output a pointer to a placer that will be set to point to the origin string inside the user-supplied url.

Returns:

0 on success or -1 on failure.

Definition at line 18 of file parse.c.

References `pl_init()`, `PLACER`, `st_cmp_ci_starts()`, `st_data_get()`, `st_empty_out()`, and `st_uchar_get()`.

Referenced by `http_response_allow_cross()`.

5.149.2.27 void http_parse_pairs (connection_t * con)

Parse all the GET and POST parameters present in an http client request, and store them with the connection.

Note:

All valid field parameters will be stored in the connection's `http.pairs` object.

Parameters:

con a pointer to the connection object generating the http requesting being processed.

Returns:

This function returns no value.

Definition at line 73 of file parse.c.

References `count`, `data`, `http_data_free()`, `HTTP_DATA_GET`, `HTTP_DATA_POST`, `http_data_value_parse()`, `HTTP_ERROR_500`, `inx_alloc()`, `M_INX_LINKED`, `pl_init()`, `PLACER`, `placer_t`, `st_char_get()`, `st_length_get()`, `st_search_cs()`, `tok_get_count_st()`, and `tok_get_st()`.

Referenced by `http_requeue()`, and `http_response()`.

5.149.2.28 void http_print_301 (connection_t * con, chr_t * location, int_t tls)

errors.c errors.c

Note:

SSL downgrades are not possible; in order for an ssl upgrade, `magma.web.ssl_redirect` must first be set.

Parameters:

con the client's http connection handle.

location the url to which the client will be redirected.

tls if 1, direct to `https://` else direct to `http://` address.

Returns:

This function returns no value.

LOW: This function should probably move to the response file and be updated to use the cookie/xss helper functions.

Definition at line 18 of file errors.c.

References `con_print()`, `con_secure()`, `HTTP_COMPLETE`, `http_print_403()`, `http_print_500()`, `log_pedantic`, `magma`, `st_char_get()`, `st_cleanup`, `st_dupe()`, `st_free()`, `st_length_int()`, `st_merge`, `magma_t::tls_redirect`, and `magma_t::web`.

Referenced by `contact_process()`, `portal_endpoint()`, `portal_process()`, `portal_upload()`, `register_process()`, and `teacher_process()`.

5.149.2.29 void http_print_400 (connection_t * con)

Return an HTTP 400 bad client request response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 80 of file errors.c.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_requeue()`.

5.149.2.30 void http_print_403 (connection_t * con)

Return an HTTP 403 access denied response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 94 of file errors.c.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_print_301()`, and `http_requeue()`.

5.149.2.31 void http_print_404 (connection_t * con)

Return an HTTP 404 location not found response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 108 of file errors.c.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_requeue()`.

5.149.2.32 void http_print_405 (connection_t * con)

Return an HTTP 405 method not allowed response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 122 of file errors.c.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_requeue()`.

5.149.2.33 void http_print_500 (connection_t * con)

Return an HTTP 500 internal server error response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 136 of file errors.c.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, `log_options`, `M_LOG_CRITICAL`, `M_LOG_STACK_TRACE`, and `PLACER`.

Referenced by `contact_print_message()`, `http_print_301()`, `http_requeue()`, `portal_print_login()`, `register_print_captcha()`, `register_print_message()`, `register_print_step3()`, `statistics_process()`, `teacher_print_form()`, and `teacher_print_message()`.

5.149.2.34 void http_print_500_log (connection_t * con, chr_t * logmsg)

Return an HTTP 500 internal server error response to the client, with additional logging information.

Parameters:

con the client's http connection handle.

logmsg a pointer to a null-terminated string with an message describing the cause of the http 500 error.

Returns:

This function returns no value.

Definition at line 154 of file errors.c.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, `log_options`, `M_LOG_CRITICAL`, and `PLACER`.

Referenced by `contact_print_message()`, `register_print_step1()`, and `register_print_step2()`.

5.149.2.35 void http_print_501 (connection_t * con)

Return an HTTP 501 method not implemented response to the client.

Parameters:

con the client's http connection handle.

Returns:

This function returns no value.

Definition at line 169 of file errors.c.

References `con_write_st()`, `HTTP_CONNECTION_CLOSE`, `http_response_header()`, and `PLACER`.

Referenced by `http_requeue()`.

5.149.2.36 void http_process (connection_t * con)

Process data sent by an http client.

Note:

This is performed in two stages: first read the http method with [http_parse_method\(\)](#), then transfer control to [http_parse_header\(\)](#). If a failure occurs reading a line of data, or too many protocol violations occur, [http_close\(\)](#) is called to drop the connection.

Parameters:

con the connection object from which to read data.

Returns:

This function returns no value.

Definition at line 137 of file http.c.

References [con_read_line\(\)](#), [enqueue\(\)](#), [http_close\(\)](#), [http_parse_header\(\)](#), [HTTP_PARSE_HEADER](#), [http_parse_method\(\)](#), [http_process\(\)](#), [HTTP_READY](#), [http_requeue\(\)](#), [log_pedantic](#), [pl_empty\(\)](#), and [requeue\(\)](#).

Referenced by [http_init\(\)](#), [http_process\(\)](#), and [http_requeue\(\)](#).

5.149.2.37 void http_requeue (connection_t * con)

The main http server requeue entry point state machine for processing client data.

Returns:

This function returns no value.

Definition at line 24 of file http.c.

References [con_status\(\)](#), [enqueue\(\)](#), [http_body\(\)](#), [http_close\(\)](#), [HTTP_CLOSE](#), [HTTP_COMPLETE](#), [HTTP_ERROR_400](#), [HTTP_ERROR_403](#), [HTTP_ERROR_404](#), [HTTP_ERROR_405](#), [HTTP_ERROR_500](#), [HTTP_ERROR_501](#), [http_parse_pairs\(\)](#), [HTTP_PARSE_PAIRS](#), [http_print_400\(\)](#), [http_print_403\(\)](#), [http_print_404\(\)](#), [http_print_405\(\)](#), [http_print_500\(\)](#), [http_print_501\(\)](#), [http_process\(\)](#), [HTTP_READ_BODY](#), [http_requeue\(\)](#), [HTTP_RESPOND](#), [http_response\(\)](#), [http_session_reset\(\)](#), [requeue\(\)](#), and [status](#).

Referenced by [http_process\(\)](#), and [http_requeue\(\)](#).

5.149.2.38 void http_response (connection_t * con)

[response.c](#) [response.c](#)

Note:

The following http methods aren't supported: PUT, DELETE, HEAD, TRACE, and CONNECT. The http server will first attempt to retrieve the requested url as a static page; otherwise the following special locations are supported: /portal, /portal/camel, /register, /contact, /report_abuse, /teacher, and /statistics.

Returns:

This function returns no value.

Definition at line 419 of file response.c.

References [magma_t::abuse](#), [magma_t::admin](#), [con_write_st\(\)](#), [magma_t::contact](#), [contact_process\(\)](#), [content](#), [HTTP_ERROR_403](#), [HTTP_ERROR_404](#), [HTTP_ERROR_405](#), [HTTP_ERROR_500](#), [HTTP_ERROR_501](#), [http_get_static\(\)](#), [HTTP_METHOD_CONNECT](#), [HTTP_METHOD_DELETE](#), [HTTP_METHOD_HEAD](#), [HTTP_METHOD_OPTIONS](#), [HTTP_METHOD_POST](#), [HTTP_METHOD_PUT](#),

HTTP_METHOD_TRACE, HTTP_METHOD_UNSUPPORTED, http_parse_pairs(), HTTP_READ_BODY, HTTP_RESPOND, http_response_header(), http_response_options(), json_api_dispatch(), magma, NULLER, PLACER, portal_endpoint(), portal_process(), portal_upload(), register_process(), magma_t::registration, http_content_t::resource, st_cmp_ci_starts(), st_cmp_cs_eq(), st_cmp_cs_starts(), st_length_get(), magma_t::statistics, statistics_process(), teacher_process(), http_content_t::type, and magma_t::web.

Referenced by http_requeue().

5.149.2.39 stringer_t* http_response_allow_cross (connection_t * con)

Definition at line 245 of file response.c.

References http_data_get(), HTTP_DATA_HEADER, http_parse_origin(), MANAGEDBUF, PLACER, placer_t, st_append, st_char_get(), st_cmp_ci_eq(), st_length_int(), st_quick(), upper_st(), and http_data_t::value.

Referenced by http_response_header(), and http_response_options().

5.149.2.40 stringer_t* http_response_connection (connection_t * con, int_t force)

Get an appropriate value for the Connection header of an http response.

Note:

If magma.http.close was set in the configuration options, the connection will be closed immediately.

Parameters:

con a pointer to the connection object of the outgoing response.

force either HTTP_CONNECTION_CLOSE, HTTP_CONNECTION_NEUTRAL, or HTTP_CONNECTION_KEEPALIVE. HTTP_CONNECTION_CLOSE will result in the connection being terminated immediately, HTTP_CONNECTION_KEEPALIVE will result in a "Connection: keep-alive" response, and HTTP_CONNECTION_NEUTRAL will not result in any output.

Returns:

a pointer to a managed string containing the http Connection header field that should be used for the response.

Definition at line 288 of file response.c.

References CONTIGUOUS, HEAP, magma_t::http, HTTP_CLOSE, HTTP_CONNECTION_CLOSE, HTTP_CONNECTION_KEEPALIVE, magma, MANAGED_T, PLACER, and st_dupe_opts().

Referenced by http_response_header(), and http_response_options().

5.149.2.41 stringer_t* http_response_cookie (connection_t * con)

Definition at line 190 of file response.c.

References con_secure(), HEAP, HTTP_COOKIE_DELETE, HTTP_COOKIE_SET, JOINED, MANAGED_T, MANAGEDBUF, PLACER, st_append, st_aprint_opts(), st_char_get(), st_length_int(), st_quick(), and time_print_gmt().

Referenced by http_response_header().

5.149.2.42 void http_response_header (connection_t * con, int_t status, stringer_t * type, size_t len)

Send a full set of http response headers to the remote client.

Note:

If the mode is HTTP_RESPOND it will be changed to HTTP_COMPLETE, to tell the http requeue function to reset the context and enqueue request processor.

Parameters:

con a pointer to the connection object across which the response will be sent.
status an integer containing the http status code for the response.
type a managed string containing the value of the Content-Type header.
len the value of the Content-Length header.

Returns:

This function returns no value.

Definition at line 364 of file response.c.

References `con_print()`, `magma_t::http`, `HTTP_COMPLETE`, `HTTP_CONNECTION_NEUTRAL`, `HTTP_RESPOND`, `http_response_allow_cross()`, `http_response_connection()`, `http_response_cookie()`, `http_response_status()`, `magma`, `MANAGEDBUF`, `mm_wipe()`, `st_char_get()`, `st_cleanup`, `st_length_int()`, and `time_print_gmt()`.

Referenced by `api_response()`, `contact_print_form()`, `contact_print_message()`, `http_print_400()`, `http_print_403()`, `http_print_404()`, `http_print_405()`, `http_print_500()`, `http_print_500_log()`, `http_print_501()`, `http_response()`, `portal_endpoint()`, `portal_endpoint_attachments_progress()`, `portal_endpoint_error()`, `portal_endpoint_response()`, `portal_endpoint_search()`, `portal_print_login()`, `register_print_message()`, `register_print_step1()`, `register_print_step2()`, `register_print_step3()`, `statistics_process()`, and `teacher_print_form()`.

5.149.2.43 void http_response_options (connection_t * con)

Return a response to an http OPTIONS request.

Parameters:

con the client connection which made the OPTIONS request.

Returns:

This function returns no value.

LOW: I couldn't find a definitive answer about whether the OPTION response should always return a Content-Type of text/plain or use the Content-Type associated with the location provided in the request.

Definition at line 316 of file response.c.

References `build_stamp()`, `build_version()`, `con_print()`, `magma_t::http`, `HTTP_COMPLETE`, `HTTP_CONNECTION_NEUTRAL`, `HTTP_RESPOND`, `http_response_allow_cross()`, `http_response_connection()`, `magma`, `MANAGEDBUF`, `mm_wipe()`, `st_char_get()`, `st_cleanup`, `st_length_int()`, and `time_print_gmt()`.

Referenced by `http_response()`.

5.149.2.44 chr_t* http_response_status (int_t status)

Get a descriptive string for a numerical http status code.

Parameters:

status the value of the http status code to be looked up.

Returns:

NULL on failure, or a pointer to a null-terminated string containing a description of the specified http status code on success.

Definition at line 15 of file response.c.

References `log_pedantic`.

Referenced by `http_response_header()`.

5.149.2.45 void http_session_destroy (connection_t * con)

sessions.c sessions.c

Parameters:

con the connection object to have its http session destroyed.

Returns:

This function returns no value.

Definition at line 77 of file sessions.c.

References `http_session_reset()`.

Referenced by `con_destroy()`.

5.149.2.46 void http_session_reset (connection_t * con)

Reset a client http connection to its original, uninitialized state.

Parameters:

con the connection object to have its http session state reset.

Returns:

This function returns no value.

Definition at line 15 of file sessions.c.

References `HTTP_CLOSE`, `HTTP_CONNECTION_CLOSE`, `HTTP_CONNECTION_NEUTRAL`, `HTTP_MERGED`, `HTTP_METHOD_NONE`, `HTTP_PORTAL`, `HTTP_READY`, `inx_cleanup()`, `json_decref_d`, `sess_release()`, and `st_cleanup`.

Referenced by `http_requeue()`, and `http_session_destroy()`.

5.149.2.47 bool_t multipart_get_boundary (connection_t * con, placer_t * output)

Get the boundary delimiter for a request by a connection specifying a Content-Type of multipart/form-data.

Parameters:

con a pointer to the connection object of the client making the http request.

output a pointer to the address of a managed string that will receive a copy of the value of the boundary string on success.

Returns:

true on success or false on failure.

Definition at line 373 of file parse.c.

References `get_header_opt()`, `get_header_value_noopt()`, `http_data_get()`, `HTTP_DATA_HEADER`, `NULLER`, `pl_empty()`, `placer_t`, and `http_data_t::value`.

Referenced by `portal_upload()`.

5.150 src/network/imap.h File Reference

Data Structures

- struct [imap_folder_status_t](#)
- struct [imap_fetch_dataitems_t](#)
- struct [imap_fetch_response_t](#)

Typedefs

- typedef [array_t](#) [imap_arguments_t](#)

Functions

- struct [__attribute__](#) ((packed))

Variables

- [imap_session_t](#)

5.150.1 Typedef Documentation

5.150.1.1 typedef array_t imap_arguments_t

Definition at line 11 of file [imap.h](#).

5.150.2 Function Documentation

5.150.2.1 struct __attribute__ ((packed)) [read]

Definition at line 28 of file [imap.h](#).

References [command](#), [meta_user_t](#), [__attribute__::session_state](#), [__attribute__::user](#), [__attribute__::username](#), and [__attribute__::usernum](#).

5.150.3 Variable Documentation

5.150.3.1 imap_session_t

Definition at line 34 of file [imap.h](#).

Referenced by [__attribute__\(\)](#).

5.151 src/servers/imap/imap.h File Reference

Defines

- #define [IMAP_ARRAY_RECURSION_LIMIT](#) 16
- #define [IMAP_SEARCH_RECURSION_LIMIT](#) 16
- #define [IMAP_FOLDER_RECURSION_LMIIT](#) 16
- #define [IMAP_ARGUMENT_TYPE_EMPTY](#) 0
- #define [IMAP_ARGUMENT_TYPE_ARRAY](#) 1
- #define [IMAP_ARGUMENT_TYPE_ASTRING](#) 2
- #define [IMAP_ARGUMENT_TYPE_QSTRING](#) 3
- #define [IMAP_ARGUMENT_TYPE_NSTRING](#) 4
- #define [IMAP_ARGUMENT_TYPE_LITERAL](#) 5
- #define [IMAP_FETCH_BODY_HEADER_FIELDS_NOT](#) 1
- #define [IMAP_FETCH_BODY_HEADER_FIELDS](#) 2
- #define [IMAP_FETCH_BODY_HEADER](#) 3
- #define [IMAP_FETCH_BODY_TEXT](#) 4
- #define [IMAP_FETCH_BODY_MIME](#) 5
- #define [IMAP_FETCH_BODY_PART](#) 6
- #define [IMAP_FLAG_SILENT](#) 1
- #define [IMAP_FLAG_ADD](#) 2
- #define [IMAP_FLAG_REMOVE](#) 4
- #define [IMAP_FLAG_REPLACE](#) 8

Functions

- [int_t imap_compare](#) (const void *compare, const void *command)
commands.c
- void [imap_process](#) ([connection_t](#) *con)
Perform client command processing on an established imap session.
- void [imap_requeue](#) ([connection_t](#) *con)
Requeue an imap connection for processing, or log it out if there was an error or excess of protocol violations.
- void [imap_sort](#) (void)
Sort the IMAP command table to be ready for binary searches.
- [imap_fetch_response_t](#) * [imap_fetch_response_add](#) ([imap_fetch_response_t](#) *response, [stringer_t](#) *key, [stringer_t](#) *value)
fetch_response.c
- void [imap_fetch_response_free](#) ([imap_fetch_response_t](#) *response)
- [inx_t](#) * [imap_duplicate_messages](#) ([inx_t](#) *messages)
fetch.c
- [imap_fetch_response_t](#) * [imap_fetch_body](#) ([array_t](#) *outer, [array_t](#) *partial, [connection_t](#) *con, [meta_message_t](#) *meta, [mail_message_t](#) **message, [stringer_t](#) **header, [imap_fetch_response_t](#) *output)
- [stringer_t](#) * [imap_fetch_body_header](#) ([placer_t](#) header, [imap_arguments_t](#) *array, [int_t](#) not)
- [stringer_t](#) * [imap_fetch_body_mime](#) ([placer_t](#) header)
- [mail_mime_t](#) * [imap_fetch_body_part](#) ([mail_message_t](#) *message, [placer_t](#) portion)
- [placer_t](#) [imap_fetch_body_portion](#) ([stringer_t](#) *part)
- [stringer_t](#) * [imap_fetch_body_tag](#) ([stringer_t](#) *tag, [array_t](#) *items)
- [stringer_t](#) * [imap_fetch_bodystructure](#) ([mail_mime_t](#) *mime)

- `stringer_t * imap_fetch_envelope (stringer_t *header)`
- `void imap_fetch_free_items (imap_fetch_dataitems_t *items)`
- `imap_fetch_response_t * imap_fetch_message (connection_t *con, meta_message_t *meta, imap_fetch_dataitems_t *items)`
- `int_t imap_fetch_parse_partial (stringer_t *partial, size_t *start, size_t *length)`
- `stringer_t * imap_fetch_return_header (connection_t *con, meta_message_t *meta, mail_message_t **message, stringer_t **header, imap_fetch_response_t *output)`
- `mail_message_t * imap_fetch_return_message (connection_t *con, meta_message_t *meta, mail_message_t **message, stringer_t **header, imap_fetch_response_t *output)`
- `mail_mime_t * imap_fetch_return_mime (connection_t *con, meta_message_t *meta, mail_message_t **message, stringer_t **header, imap_fetch_response_t *output)`
- `stringer_t * imap_fetch_return_text (connection_t *con, meta_message_t *meta, mail_message_t **message, stringer_t **header, imap_fetch_response_t *output)`
- `inx_t * imap_narrow_messages (inx_t *messages, uint64_t selected, stringer_t *range, int_t uid)`
- `imap_fetch_dataitems_t * imap_parse_dataitems (imap_arguments_t *arguments)`
- `int_t imap_valid_sequence (stringer_t *range)`
- `int_t imap_flag_action (stringer_t *string)`

flags.c

- `uint32_t imap_flag_parse (void *ptr, int_t type)`
- `uint32_t imap_get_flag (stringer_t *string)`
- `void imap_update_flags (meta_user_t *user, inx_t *messages, uint64_t foldernum, int_t action, uint32_t flags)`
- `int_t imap_count_folder_levels (stringer_t *name)`

folders.c

- `int_t imap_folder_compare (stringer_t *name, stringer_t *compare)`
- `int_t imap_folder_create (uint64_t usernum, inx_t *folders, stringer_t *name)`

Create a new imap folder.

- `stringer_t * imap_folder_name_escaped (inx_t *folders, meta_folder_t *active)`

Get an imap folder's escaped name.

- `int_t imap_folder_remove (uint64_t usernum, inx_t *folders, inx_t *messages, stringer_t *name)`

Remove an imap folder, both in memory and on the database.

- `int_t imap_folder_rename (uint64_t usernum, inx_t *folders, stringer_t *original, stringer_t *rename)`

Rename an imap folder.

- `int_t imap_folder_status (inx_t *folders, inx_t *messages, stringer_t *name, imap_folder_status_t *status)`

Get the status of a folder.

- `inx_t * imap_narrow_folders (inx_t *folders, stringer_t *reference, stringer_t *mailbox)`
- `uint64_t imap_next_folder_order (inx_t *folders, uint64_t parent)`

Get the next order value for an imap folder.

- `bool_t imap_valid_folder_name (stringer_t *name)`

Check to see if a name is a valid imap folder name.

- `void imap_append (connection_t *con)`

imap.c

- `void imap_capability (connection_t *con)`

Display the capability string for the IMAP server.

- void [imap_check](#) ([connection_t](#) *con)
- void [imap_close](#) ([connection_t](#) *con)
- void [imap_copy](#) ([connection_t](#) *con)
- void [imap_create](#) ([connection_t](#) *con)

Create a new IMAP folder in response to the IMAP "CREATE" command.

- void [imap_delete](#) ([connection_t](#) *con)

Handle the IMAP "DELETE" command and delete the specified IMAP folder.

- void [imap_examine](#) ([connection_t](#) *con)
- void [imap_expunge](#) ([connection_t](#) *con)
- void [imap_fetch](#) ([connection_t](#) *con)
- void [imap_id](#) ([connection_t](#) *con)
- void [imap_idle](#) ([connection_t](#) *con)
- void [imap_init](#) ([connection_t](#) *con)

The main IMAP entry point for all inbound client connections, as dispatched by the generic protocol handler (display banner greeting).

- void [imap_invalid](#) ([connection_t](#) *con)

Respond to an invalid IMAP command from a client.

- void [imap_list](#) ([connection_t](#) *con)
- void [imap_login](#) ([connection_t](#) *con)

Attempt to perform a user login on an IMAP client connection.

- void [imap_logout](#) ([connection_t](#) *con)

Terminate an IMAP session gracefully with a "BYE" message and destroy the underlying connection.

- void [imap_lsub](#) ([connection_t](#) *con)
- void [imap_noop](#) ([connection_t](#) *con)
- void [imap_rename](#) ([connection_t](#) *con)
- void [imap_search](#) ([connection_t](#) *con)
- void [imap_select](#) ([connection_t](#) *con)
- void [imap_starttls](#) ([connection_t](#) *con)

Create a secure connection for an IMAP session.

- void [imap_status](#) ([connection_t](#) *con)
- void [imap_store](#) ([connection_t](#) *con)
- void [imap_subscribe](#) ([connection_t](#) *con)
- void [imap_unsubscribe](#) ([connection_t](#) *con)
- [int_t](#) [imap_append_message](#) ([connection_t](#) *con, [meta_folder_t](#) *folder, [uint32_t](#) flags, [stringer_t](#) *message, [uint64_t](#) *outnum)

messages.c

- [int_t](#) [imap_message_copier](#) ([connection_t](#) *con, [meta_message_t](#) *message, [uint64_t](#) target, [uint64_t](#) *outnum)
- [int_t](#) [imap_message_expunge](#) ([connection_t](#) *con, [meta_message_t](#) *message)
- [stringer_t](#) * [imap_build_array](#) ([chr_t](#) *format,...)

output.c

- [stringer_t](#) * [imap_build_array_isliteral](#) ([placer_t](#) data)
- [stringer_t](#) * [imap_parse_address](#) ([stringer_t](#) *address)

parse_address.c

- [placer_t](#) [imap_parse_address_breaker](#) ([stringer_t](#) *address, [uint32_t](#) part)
- [stringer_t](#) * [imap_parse_address_part](#) ([placer_t](#) input)

- `void imap_parse_address_put (stringer_t *buffer, chr_t c)`
LOW: Do we need an `imap_parse_address_group()` function?
- `int_t imap_command_parser (connection_t *con)`
parse.c
- `imap_arguments_t * imap_get_ar_ar (imap_arguments_t *array, size_t element)`
Return the value of a specified object in an array as an imap arguments array.
- `void * imap_get_ptr (imap_arguments_t *array, size_t element)`
Return the value of a specified object in an array as a generic pointer.
- `stringer_t * imap_get_st_ar (imap_arguments_t *array, size_t element)`
Return the value of a specified object in an array as a managed string.
- `int_t imap_get_type_ar (imap_arguments_t *array, size_t element)`
TODO: The logic here is just plain ugly. Specifically the logic used to read from the network and advance the buffers.
- `int_t imap_parse_arguments (connection_t *con, chr_t **start, size_t *length)`
Parse a chunk of input into an array of IMAP arguments.
- `int_t imap_parse_array (int_t recursion, connection_t *con, imap_arguments_t **array, chr_t **start, size_t *length)`
Extract the contents of an array argument and advance the position of the parser stream.
- `int_t imap_parse_astring (stringer_t **output, chr_t **start, size_t *length)`
Extract the contents of an atomic string and advance the position of the parser stream.
- `int_t imap_parse_literal (connection_t *con, stringer_t **output, chr_t **start, size_t *length)`
Extract the contents of a literal string and advance the position of the parser stream.
- `int_t imap_parse_nstring (stringer_t **output, chr_t **start, size_t *length, chr_t type)`
- `int_t imap_parse_qstring (stringer_t **output, chr_t **start, size_t *length)`
Extract the contents of a quoted string and advance the position of the parser stream.
- `stringer_t * imap_range_build (size_t length, uint64_t *numbers)`
range.c
- `int_t imap_search_flag (uint32_t status, uint32_t flag, int_t has)`
search.c
- `inx_t * imap_search_messages (connection_t *con)`
- `int_t imap_search_messages_body (connection_t *con, meta_user_t *user, mail_message_t **data, meta_message_t *active, stringer_t *value)`
- `int_t imap_search_messages_date (connection_t *con, meta_user_t *user, mail_message_t **data, stringer_t **header, meta_message_t *active, stringer_t *date, int_t internal, int_t expected)`
- `int_t imap_search_messages_date_compare (stringer_t *one, stringer_t *two)`
- `int_t imap_search_messages_header (connection_t *con, meta_user_t *user, mail_message_t **data, stringer_t **header, meta_message_t *active, stringer_t *field, stringer_t *value)`
- `int_t imap_search_messages_inner (connection_t *con, meta_user_t *user, mail_message_t **message, stringer_t **header, meta_message_t *current, imap_arguments_t *array, unsigned recursion)`
- `int_t imap_search_messages_range (meta_message_t *active, stringer_t *range, int_t uid)`
- `int_t imap_search_messages_size (meta_message_t *active, stringer_t *value, int_t expected)`
- `int_t imap_search_messages_text (connection_t *con, meta_user_t *user, mail_message_t **data, meta_message_t *active, stringer_t *value)`

- void `imap_session_destroy` (`connection_t *con`)

sessions.c

- int `imap_session_update` (`connection_t *con`)

Returns 0 if the selected folder wasn't modified, or 1 if things changed and the updated status should be sent to the client, and a -1 is used to indicate the update check encountered a problem and should be retried later.

5.151.1 Define Documentation

5.151.1.1 #define IMAP_ARGUMENT_TYPE_ARRAY 1

Definition at line 17 of file `imap.h`.

Referenced by `imap_append()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_fetch()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_flag_parse()`, `imap_get_ar_ar()`, `imap_get_st_ar()`, `imap_id()`, `imap_list()`, `imap_login()`, `imap_lsub()`, `imap_parse_arguments()`, `imap_parse_array()`, `imap_parse_dataitems()`, `imap_rename()`, `imap_search_messages_inner()`, `imap_select()`, `imap_status()`, `imap_store()`, and `imap_subscribe()`.

5.151.1.2 #define IMAP_ARGUMENT_TYPE_ASTRING 2

Definition at line 18 of file `imap.h`.

Referenced by `imap_parse_arguments()`.

5.151.1.3 #define IMAP_ARGUMENT_TYPE_EMPTY 0

Definition at line 16 of file `imap.h`.

Referenced by `imap_get_type_ar()`.

5.151.1.4 #define IMAP_ARGUMENT_TYPE_LITERAL 5

Definition at line 21 of file `imap.h`.

Referenced by `imap_append()`, `imap_parse_arguments()`, and `imap_parse_array()`.

5.151.1.5 #define IMAP_ARGUMENT_TYPE_NSTRING 4

Definition at line 20 of file `imap.h`.

Referenced by `imap_parse_array()`.

5.151.1.6 #define IMAP_ARGUMENT_TYPE_QSTRING 3

Definition at line 19 of file `imap.h`.

Referenced by `imap_parse_arguments()`, and `imap_parse_array()`.

5.151.1.7 #define IMAP_ARRAY_RECURSION_LIMIT 16

Definition at line 11 of file `imap.h`.

Referenced by `imap_parse_array()`.

5.151.1.8 #define IMAP_FETCH_BODY_HEADER 3

Definition at line 26 of file imap.h.

5.151.1.9 #define IMAP_FETCH_BODY_HEADER_FIELDS 2

Definition at line 25 of file imap.h.

5.151.1.10 #define IMAP_FETCH_BODY_HEADER_FIELDS_NOT 1

Definition at line 24 of file imap.h.

5.151.1.11 #define IMAP_FETCH_BODY_MIME 5

Definition at line 28 of file imap.h.

5.151.1.12 #define IMAP_FETCH_BODY_PART 6

Definition at line 29 of file imap.h.

5.151.1.13 #define IMAP_FETCH_BODY_TEXT 4

Definition at line 27 of file imap.h.

5.151.1.14 #define IMAP_FLAG_ADD 2

Definition at line 33 of file imap.h.

Referenced by `imap_flag_action()`, and `imap_update_flags()`.

5.151.1.15 #define IMAP_FLAG_REMOVE 4

Definition at line 34 of file imap.h.

Referenced by `imap_flag_action()`, and `imap_update_flags()`.

5.151.1.16 #define IMAP_FLAG_REPLACE 8

Definition at line 35 of file imap.h.

Referenced by `imap_flag_action()`, and `imap_update_flags()`.

5.151.1.17 #define IMAP_FLAG_SILENT 1

Definition at line 32 of file imap.h.

Referenced by `imap_flag_action()`, and `imap_store()`.

5.151.1.18 #define IMAP_FOLDER_RECURSION_LMIT 16

Definition at line 13 of file `imap.h`.

Referenced by `contact_folder_create()`, `imap_count_folder_levels()`, `imap_create()`, `imap_folder_create()`, `imap_folder_rename()`, `imap_rename()`, and `imap_subscribe()`.

5.151.1.19 #define IMAP_SEARCH_RECURSION_LIMIT 16

Definition at line 12 of file `imap.h`.

Referenced by `imap_search_messages_inner()`.

5.151.2 Function Documentation**5.151.2.1 void imap_append (connection_t * con)**

`imap.c`

BUG: If the user is over their quota check whether rollout is enabled and if so. If enabled make room for the appended message using the rollout logic.

Definition at line 1462 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `imap_append_message()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `IMAP_ARGUMENT_TYPE_LITERAL`, `imap_flag_parse()`, `imap_get_ptr()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `MAIL_STATUS_EMPTY`, `MAIL_STATUS_RECENT`, `meta_folder_t`, `meta_folders_by_name()`, `meta_message_t`, `META_USER_OVERQUOTA`, `meta_user_unlock()`, `meta_user_wlock()`, `st_char_get()`, and `st_length_int()`.

5.151.2.2 int_t imap_append_message (connection_t * con, meta_folder_t * folder, uint32_t flags, stringer_t * message, uint64_t * outnum)

`messages.c`

Definition at line 10 of file `messages.c`.

References `magma_t::active`, `inx_append()`, `log_pedantic`, `M_TYPE_UINT64`, `magma`, `MAIL_STATUS_APPENDED`, `MAIL_STATUS_RECENT`, `mail_store_message()`, `meta_message_t`, `meta_messages_update_sequences()`, `META_USER_ENCRYPT_DATA`, `mm_alloc()`, `mm_free()`, `OBJECT_MESSAGES`, `serial_get()`, `serial_increment()`, `st_char_get()`, `st_length_get()`, `st_length_int()`, `magma_t::storage`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_append()`.

5.151.2.3 stringer_t* imap_build_array (chr_t * format, ...)

`output.c`

Definition at line 37 of file `output.c`.

References `imap_build_array_isliteral()`, `length`, `log_error`, `log_pedantic`, `ns_length_get()`, `number`, `pl_data_get()`, `pl_empty()`, `pl_init()`, `pl_length_get()`, `placer_t`, `st_alloc()`, `st_char_get()`, `st_free()`, `st_length_get()`, and `st_length_set()`.

Referenced by `imap_fetch_bodystructure()`, `imap_fetch_envelope()`, `imap_parse_address()`, and `imap_parse_address_part()`.

5.151.2.4 stringer_t* imap_build_array_isliteral (placer_t data)

Definition at line 10 of file `output.c`.

References `length`, `pl_data_get()`, `pl_empty()`, `pl_length_get()`, and `st_merge`.

Referenced by `imap_build_array()`, and `imap_fetch_bodystructure()`.

5.151.2.5 `void imap_capability (connection_t * con)`

Display the capability string for the IMAP server.

Note:

This string always needs to stay in sync with the banner greeting.

Returns:

This function returns no value.

Definition at line 1831 of file `imap.c`.

References `con_print()`, `con_secure()`, `st_char_get()`, and `st_length_int()`.

5.151.2.6 `void imap_check (connection_t * con)`

Definition at line 297 of file `imap.c`.

References `con_print()`, `imap_session_update()`, `st_char_get()`, and `st_length_int()`.

5.151.2.7 `void imap_close (connection_t * con)`

Definition at line 1087 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `imap_message_expunge()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_RECENT`, `meta_message_t`, `meta_messages_update_sequences()`, `meta_user_rlock()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_MESSAGES`, `serial_get()`, `serial_increment()`, `st_char_get()`, `st_length_int()`, `user_lock()`, and `user_unlock()`.

5.151.2.8 `int_t imap_command_parser (connection_t * con)`

`parse.c` `parse.c`

Note:

This function updates the protocol-specific IMAP structure with parsed values for the session's tag, command, and arguments fields. Special handling is performed for any command that is preceded by a "UID" prefix.

Parameters:

con the IMAP client connection issuing the command.

Returns:

1 on success or < 0 on error. -1: the tag could not be read. -2: the IMAP command could not be read. -3: the arguments to the IMAP command could not be read.

Definition at line 886 of file `parse.c`.

References `ar_free()`, `magma_t::imap`, `imap_command_log_safe()`, `imap_parse_arguments()`, `imap_parse_astring()`, `length`, `magma_t::log`, `log_pedantic`, `magma`, `PLACER`, `st_cleanup`, `st_cmp_ci_eq()`, `st_data_get()`, `st_free()`, and `st_length_get()`.

Referenced by `imap_process()`.

5.151.2.9 int_t imap_compare (const void * *compare*, const void * *command*)

commands.c commands.c

Note:

This is an internal function used to sort imap commands and search for them.

Parameters:

compare a pointer to the first command to be compared.

command a pointer to the second command to be compared.

Returns:

-1 if *compare* < *command*, 1 if *command* < *compare*, or 0 if the two commands are equal.

Definition at line 18 of file commands.c.

References command_t, PLACER, st_cmp_ci_eq(), and st_cmp_ci_starts().

Referenced by imap_process(), and imap_sort().

5.151.2.10 void imap_copy (connection_t * *con*)

Definition at line 1296 of file imap.c.

References ar_length_get(), con_print(), count, IMAP_ARGUMENT_TYPE_ARRAY, imap_get_st_ar(), imap_get_type_ar(), imap_message_copier(), imap_narrow_messages(), imap_range_build(), imap_valid_sequence(), inx_count(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_free(), inx_t, log_pedantic, MAIL_STATUS_RECENT, meta_folder_t, meta_folders_by_name(), meta_message_t, meta_user_unlock(), meta_user_wlock(), mm_alloc(), mm_free(), st_char_get(), st_length_get(), st_length_int(), user_lock(), and user_unlock().

5.151.2.11 int_t imap_count_folder_levels (stringer_t * *name*)

folders.c folders.c

Note:

The smallest possible node level value is 1.

Parameters:

name a managed string containing the name of the imap folder.

Returns:

0 on failure, or the number of node levels indicated by the specified imap folder name, on success.

Definition at line 218 of file folders.c.

References IMAP_FOLDER_RECURSION_LMIIT, log_pedantic, and st_empty_out().

Referenced by imap_folder_create(), and imap_folder_rename().

5.151.2.12 void imap_create (connection_t * *con*)

Create a new IMAP folder in response to the IMAP "CREATE" command.

See also:

[imap_folder_create\(\)](#)

Parameters:

con the connection across which the folder creation request was made.

Returns:

This function returns no value.

Definition at line 471 of file imap.c.

References [ar_length_get\(\)](#), [con_print\(\)](#), [FOLDER_LENGTH_LIMIT](#), [IMAP_ARGUMENT_TYPE_ARRAY](#), [imap_folder_create\(\)](#), [IMAP_FOLDER_RECURSION_LMIIT](#), [imap_get_st_ar\(\)](#), [imap_get_type_ar\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_FOLDERS](#), [serial_get\(\)](#), [serial_increment\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

5.151.2.13 void imap_delete (connection_t * con)

Handle the IMAP "DELETE" command and delete the specified IMAP folder.

See also:

[imap_folder_remove\(\)](#)

Parameters:

con a pointer to the connection object of the IMAP session generating the delete request.

Returns:

This function returns no value.

Definition at line 538 of file imap.c.

References [ar_length_get\(\)](#), [con_print\(\)](#), [IMAP_ARGUMENT_TYPE_ARRAY](#), [imap_folder_remove\(\)](#), [imap_get_st_ar\(\)](#), [imap_get_type_ar\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_FOLDERS](#), [serial_get\(\)](#), [serial_increment\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

5.151.2.14 inx_t* imap_duplicate_messages (inx_t * messages)

[fetch.c](#) [fetch.c](#)

See also:

[meta_message_dupe\(\)](#)

Parameters:

messages a collection of meta message objects to be duplicated.

Returns:

the copied collection of meta messages on success, or NULL on failure.

Definition at line 1240 of file fetch.c.

References [inx_alloc\(\)](#), [inx_append\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [inx_free\(\)](#), [inx_t](#), [log_error](#), [M_INX_LINKED](#), [M_TYPE_UINT64](#), [meta_message_dupe\(\)](#), [meta_message_free\(\)](#), [meta_message_t](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [imap_fetch\(\)](#), and [imap_store\(\)](#).

5.151.2.15 void imap_examine (connection_t * con)

Definition at line 802 of file imap.c.

References `ar_length_get()`, `con_print()`, `imap_folder_status_t::first`, `imap_folder_status_t::foldernum`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_status()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `MAIL_STATUS_RECENT`, `imap_folder_status_t::messages`, `meta_message_t`, `meta_user_rlock()`, `meta_user_unlock()`, `meta_user_wlock()`, `imap_folder_status_t::recent`, `st_char_get()`, `st_length_int()`, `status`, and `imap_folder_status_t::uidnext`.

5.151.2.16 void imap_expunge (connection_t * con)

Definition at line 1197 of file imap.c.

References `ar_length_get()`, `con_print()`, `imap_message_expunge()`, `imap_session_update()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `MAIL_STATUS_DELETED`, `meta_message_t`, `meta_messages_update_sequences()`, `meta_user_rlock()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_MESSAGES`, `serial_get()`, `serial_increment()`, `st_char_get()`, `st_length_int()`, `user_lock()`, and `user_unlock()`.

5.151.2.17 void imap_fetch (connection_t * con)

Definition at line 1547 of file imap.c.

References `ar_length_get()`, `con_print()`, `con_status()`, `con_write_bl()`, `con_write_st()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_duplicate_messages()`, `imap_fetch_free_items()`, `imap_fetch_message()`, `imap_fetch_response_free()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_narrow_messages()`, `imap_parse_dataitems()`, `imap_valid_sequence()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_reset()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `items`, `imap_fetch_response_t::key`, `MAIL_STATUS_SEEN`, `meta_data_flags_add()`, `meta_message_t`, `meta_user_rlock()`, `meta_user_unlock()`, `meta_user_wlock()`, `imap_fetch_response_t::next`, `imap_fetch_dataitems_t::normal`, `OBJECT_MESSAGES`, `PLACER`, `imap_fetch_dataitems_t::rfc822`, `imap_fetch_dataitems_t::rfc822_header`, `imap_fetch_dataitems_t::rfc822_text`, `serial_get()`, `serial_increment()`, `st_char_get()`, `st_cmp_cs_eq()`, `st_length_int()`, `status`, and `imap_fetch_response_t::value`.

5.151.2.18 imap_fetch_response_t* imap_fetch_body (array_t * outer, array_t * partial, connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)

Definition at line 762 of file fetch.c.

References `ar_field_ar()`, `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_ARRAY`, `mail_mime_t::body`, `mail_mime_t::header`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_fetch_body_header()`, `imap_fetch_body_mime()`, `imap_fetch_body_part()`, `imap_fetch_body_portion()`, `imap_fetch_body_tag()`, `imap_fetch_parse_partial()`, `imap_fetch_response_add()`, `imap_fetch_response_free()`, `imap_fetch_return_header()`, `imap_fetch_return_message()`, `imap_fetch_return_mime()`, `imap_fetch_return_text()`, `imap_get_ar_ar()`, `imap_get_ptr()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `length`, `mail_destroy()`, `mail_destroy_header()`, `mail_message_t::mime`, `number`, `pl_data_get()`, `pl_empty()`, `pl_init()`, `pl_length_get()`, `pl_null()`, `PLACER`, `placer_t`, `st_char_get()`, `st_cleanup`, `st_cmp_ci_eq()`, `st_free()`, `st_import()`, `st_length_get()`, and `st_merge`.

Referenced by `imap_fetch_message()`.

5.151.2.19 stringer_t* imap_fetch_body_header (placer_t header, imap_arguments_t * array, int_t not)

Definition at line 386 of file fetch.c.

References `ar_length_get()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_st_ar()`, `imap_get_type_ar()`, `mail_header_pop()`, `mm_cmp_ci_eq()`, `number`, `pl_char_get()`, `pl_data_get()`, `pl_empty()`, `placer_t`, `st_char_get()`, `st_cleanup`, `st_length_get()`, and `st_merge`.

Referenced by `imap_fetch_body()`.

5.151.2.20 stringer_t* imap_fetch_body_mime (placer_t *header*)

Definition at line 437 of file fetch.c.

References mail_header_fetch_all(), PLACER, st_cleanup, and st_merge.

Referenced by imap_fetch_body().

5.151.2.21 mail_mime_t* imap_fetch_body_part (mail_message_t * *message*, placer_t *portion*)

Definition at line 555 of file fetch.c.

References ar_field_ptr(), mail_mime_t::children, mail_message_t::mime, number, placer_t, tok_get_count_st(), tok_get_st(), and uint32_conv_st().

Referenced by imap_fetch_body().

5.151.2.22 placer_t imap_fetch_body_portion (stringer_t * *part*)

Definition at line 512 of file fetch.c.

References length, pl_init(), pl_null(), st_char_get(), and st_empty_out().

Referenced by imap_fetch_body().

5.151.2.23 stringer_t* imap_fetch_body_tag (stringer_t * *tag*, array_t * *items*)

Definition at line 464 of file fetch.c.

References ar_length_get(), imap_get_st_ar(), number, st_cleanup, st_free(), st_import(), st_merge, and upper_st().

Referenced by imap_fetch_body().

5.151.2.24 stringer_t* imap_fetch_bodystructure (mail_mime_t * *mime*)

Definition at line 229 of file fetch.c.

References ar_field_ptr(), ar_field_st(), ar_free(), ar_length_get(), mail_mime_t::body, mail_mime_t::children, mail_mime_t::header, imap_build_array(), imap_build_array_isliteral(), imap_fetch_bodystructure(), items, length, mail_header_fetch_cleaned(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_type_group(), mail_mime_type_parameters(), mail_mime_type_sub(), ns_length_get(), pl_init(), PLACER, st_char_get(), st_cleanup, st_cmp_cs_eq(), st_data_get(), st_free(), st_import(), st_length_get(), st_merge, and upper_st().

Referenced by imap_fetch_bodystructure(), and imap_fetch_message().

5.151.2.25 stringer_t* imap_fetch_envelope (stringer_t * *header*)

Definition at line 184 of file fetch.c.

References imap_build_array(), imap_parse_address(), mail_header_fetch_cleaned(), pl_init(), pl_null(), PLACER, placer_t, st_char_get(), st_cleanup, and st_length_get().

Referenced by imap_fetch_message().

5.151.2.26 void imap_fetch_free_items (imap_fetch_dataitems_t * *items*)

Definition at line 32 of file fetch.c.

References mm_cleanup, mm_free(), imap_fetch_dataitems_t::normal, imap_fetch_dataitems_t::normal_partial, imap_fetch_dataitems_t::peek, and imap_fetch_dataitems_t::peek_partial.

Referenced by `imap_fetch()`, and `imap_parse_dataitems()`.

5.151.2.27 `imap_fetch_response_t* imap_fetch_message (connection_t * con, meta_message_t * meta, imap_fetch_dataitems_t * items)`

Definition at line 1043 of file `fetch.c`.

References `imap_fetch_dataitems_t::body`, `mail_mime_t::body`, `imap_fetch_dataitems_t::bodystructure`, `CONTIGUOUS`, `imap_fetch_dataitems_t::envelope`, `imap_fetch_dataitems_t::flags`, `HEAP`, `imap_fetch_body()`, `imap_fetch_bodystructure()`, `imap_fetch_envelope()`, `imap_fetch_response_add()`, `imap_fetch_response_free()`, `imap_fetch_return_header()`, `imap_fetch_dataitems_t::internaldate`, `log_pedantic`, `mail_destroy()`, `mail_destroy_header()`, `mail_load_message()`, `mail_mime_update()`, `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `MANAGED_T`, `mail_message_t::mime`, `imap_fetch_dataitems_t::normal`, `imap_fetch_dataitems_t::normal_partial`, `ns_length_get()`, `imap_fetch_dataitems_t::peek`, `imap_fetch_dataitems_t::peek_partial`, `PLACER`, `imap_fetch_dataitems_t::rfc822`, `imap_fetch_dataitems_t::rfc822_header`, `imap_fetch_dataitems_t::rfc822_size`, `imap_fetch_dataitems_t::rfc822_text`, `st_aprint_opts()`, `st_import()`, `st_length_get()`, `st_merge`, `mail_message_t::text`, and `imap_fetch_dataitems_t::uid`.

Referenced by `imap_fetch()`.

5.151.2.28 `int_t imap_fetch_parse_partial (stringer_t * partial, size_t * start, size_t * length)`

Definition at line 609 of file `fetch.c`.

References `int32_conv_bl()`, `number`, `st_char_get()`, and `st_length_get()`.

Referenced by `imap_fetch_body()`.

5.151.2.29 `imap_fetch_response_t* imap_fetch_response_add (imap_fetch_response_t * response, stringer_t * key, stringer_t * value)`

[fetch_response.c](#)

Definition at line 25 of file `fetch_response.c`.

References `CONTIGUOUS`, `HEAP`, `imap_fetch_response_t::key`, `log_error`, `MANAGED_T`, `mm_alloc()`, `mm_free()`, `imap_fetch_response_t::next`, `st_dupe_opts()`, `st_free()`, and `imap_fetch_response_t::value`.

Referenced by `imap_fetch_body()`, and `imap_fetch_message()`.

5.151.2.30 `void imap_fetch_response_free (imap_fetch_response_t * response)`

Definition at line 10 of file `fetch_response.c`.

References `imap_fetch_response_t::key`, `mm_free()`, `imap_fetch_response_t::next`, `st_cleanup`, and `imap_fetch_response_t::value`.

Referenced by `imap_fetch()`, `imap_fetch_body()`, `imap_fetch_message()`, `imap_fetch_return_header()`, `imap_fetch_return_message()`, `imap_fetch_return_mime()`, and `imap_fetch_return_text()`.

5.151.2.31 `stringer_t* imap_fetch_return_header (connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)`

Definition at line 685 of file `fetch.c`.

References `imap_fetch_response_free()`, `mail_destroy()`, `mail_load_header()`, `st_char_get()`, and `st_import()`.

Referenced by `imap_fetch_body()`, and `imap_fetch_message()`.

5.151.2.32 `mail_message_t* imap_fetch_return_message (connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)`

Definition at line 728 of file fetch.c.

References `imap_fetch_response_free()`, `mail_destroy()`, `mail_destroy_header()`, `mail_load_message()`, and `mail_mime_update()`.

Referenced by `imap_fetch_body()`.

5.151.2.33 `mail_mime_t* imap_fetch_return_mime (connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)`

Definition at line 742 of file fetch.c.

References `imap_fetch_response_free()`, `mail_destroy()`, `mail_destroy_header()`, `mail_load_message()`, and `mail_mime_update()`.

Referenced by `imap_fetch_body()`.

5.151.2.34 `stringer_t* imap_fetch_return_text (connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)`

Definition at line 714 of file fetch.c.

References `imap_fetch_response_free()`, `mail_destroy()`, `mail_destroy_header()`, and `mail_load_message()`.

Referenced by `imap_fetch_body()`.

5.151.2.35 `int_t imap_flag_action (stringer_t * string)`

flags.c

Definition at line 41 of file flags.c.

References `IMAP_FLAG_ADD`, `IMAP_FLAG_REMOVE`, `IMAP_FLAG_REPLACE`, `IMAP_FLAG_SILENT`, `PLACER`, and `st_cmp_ci_eq()`.

Referenced by `imap_store()`.

5.151.2.36 `uint32_t imap_flag_parse (void * ptr, int_t type)`

Definition at line 65 of file flags.c.

References `ar_length_get()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_flag()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `MAIL_STATUS_EMPTY`, and `number`.

Referenced by `imap_append()`, and `imap_store()`.

5.151.2.37 `int_t imap_folder_compare (stringer_t * name, stringer_t * compare)`

Definition at line 91 of file folders.c.

References `PLACER`, `st_char_get()`, `st_cmp_ci_eq()`, and `st_length_get()`.

Referenced by `imap_narrow_folders()`.

5.151.2.38 `int_t imap_folder_create (uint64_t usernum, inx_t * folders, stringer_t * name)`

Create a new imap folder.

Note:

If they don't already exist, any parent folders specified in the fully qualified folder name will automatically be created first.

Parameters:

userid the numerical id of the user to whom the new imap folder will belong.
folders an inx holder containing the set of folders into which the new imap folder will be inserted.
name a managed string containing the fully qualified name of the new imap folder to be created.

Returns:

1 on success or ≤ 0 on failure. 0: An invalid parameter was passed to the function, or an internal failure occurred. -1: The specified new folder name was invalid. -2: The node depth of the new folder is too large (imap folder recursion limit reached [IMAP_FOLDER_RECURSION_LMIIT]). -3: Special folder "Inbox" cannot contain subfolders. -4: Part of the folder path name exceeds 16 characters (FOLDER_LENGTH_LIMIT) after being unescaped for quotation marks. -5: The specified folder name already exists.

Definition at line 264 of file folders.c.

References CONTIGUOUS, FOLDER_LENGTH_LIMIT, HEAP, imap_count_folder_levels(), IMAP_FOLDER_RECURSION_LMIIT, imap_next_folder_order(), imap_valid_folder_name(), inx_insert(), log_pedantic, M_TYPE_UINT64, MANAGED_T, meta_data_insert_folder(), meta_folder_t, meta_folders_by_name(), mm_alloc(), mm_copy(), mm_free(), pl_data_get(), pl_length_get(), PLACER, placer_t, st_cleanup, st_cmp_ci_eq(), st_dupe(), st_dupe_opts(), st_free(), st_merge, st_replace(), tok_get_st(), multi_t::u64, and multi_t::val.

Referenced by imap_create(), imap_subscribe(), and portal_folder_mail_add().

5.151.2.39 stringer_t* imap_folder_name_escaped (inx_t * folders, meta_folder_t * active)

Get an imap folder's escaped name. TODO: Since the Portal needs access to some of these folder manipulation functions the common logic should be extracted and moved into the folder object interface.

Parameters:

folders an inx holder containing the ancestor folders of the specified imap folder.
active a pointer to the meta folder object of the folder to have its name escaped.

Returns:

NULL on failure, or a pointer to a managed string containing the escaped name of the specified imap folder on success.

Definition at line 19 of file folders.c.

References meta_folders_name(), PLACER, st_free(), st_merge, and st_replace().

Referenced by imap_list(), and imap_lsub().

5.151.2.40 int_t imap_folder_remove (uint64_t userid, inx_t * folders, inx_t * messages, stringer_t * name)

Remove an imap folder, both in memory and on the database.

Note:

Any child messages of the specified folder will be deleted, but if it has child folders, the folder will not be deleted. The function also protects the special "Inbox" folder from deletion.

Parameters:

userid the numerical id of the user that owns the imap folder to be deleted.
folders an inx holder containing a collection of folders for looking up the specified imap folder by name.

messages an inx holder containing a collection of messages for looking up the contents of the specified imap folder.

name a managed string containing the name of the imap folder to be deleted.

Returns:

<= 0 on failure, or 1 on success. 0: General failure or if there was an error removing the folder from the database. -1: The specified folder name was invalid. -2: Removal failed because the "Inbox" folder was specified. -3: The specified folder could not be found.

Definition at line 412 of file folders.c.

References `imap_valid_folder_name()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_delete()`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `mail_remove_message()`, `meta_data_delete_folder()`, `meta_folder_t`, `meta_folders_by_name()`, `meta_folders_children()`, `meta_message_t`, `PLACER`, `placer_t`, `st_cmp_ci_eq()`, `tok_get_st()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_delete()`, and `portal_folder_mail_remove()`.

5.151.2.41 `int_t imap_folder_rename(uint64_t usernum, inx_t *folders, stringer_t *original, stringer_t *rename)`

Rename an imap folder.

Note:

Any parent folder components of the folder's fully qualified name will be created if they do not already exist.

Parameters:

usernum the numerical id of the user requesting the folder renaming.

folders an inx holder containing the folders to be searched for the folder specified to be renamed.

original a managed string containing the original name of the folder to be renamed.

rename a managed string containing the new name of the specified folder.

Returns:

1 on success or 0 or less on failure. 0: One of the parameters passed to the function was invalid, or there was a memory allocation failure. -1: Either the original or new folder name was invalid. -2: Was unable to rename the "Inbox" folder. -3: The specified imap folder did not exist. -4: Either the original or new name exceeded the imap folder recursion limit. -5: A folder already exists with the new folder name. -6: A segment of the folder name was larger than `FOLDER_LENGTH_LIMIT` (16 bytes).

Definition at line 488 of file folders.c.

References `CONTIGUOUS`, `FOLDER_LENGTH_LIMIT`, `HEAP`, `imap_count_folder_levels()`, `IMAP_FOLDER_RECURSION_LMIIT`, `imap_next_folder_order()`, `imap_valid_folder_name()`, `inx_insert()`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `MANAGED_T`, `meta_data_insert_folder()`, `meta_data_update_folder_name()`, `meta_folder_t`, `meta_folders_by_name()`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `mm_wipe()`, `pl_data_get()`, `pl_length_get()`, `PLACER`, `placer_t`, `st_cleanup`, `st_cmp_ci_eq()`, `st_dupe()`, `st_dupe_opts()`, `st_free()`, `st_merge`, `st_replace()`, `tok_get_st()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_rename()`, and `portal_endpoint_folders_rename()`.

5.151.2.42 `int_t imap_folder_status(inx_t *folders, inx_t *messages, stringer_t *name, imap_folder_status_t *status)`

Get the status of a folder.

Note:

This function will count the number of messages in a folder, as well as the number of messages marked recent or unseen, as well as the numerical id of the first message in the folder and the `UIDNEXT` of the specified folder.

Parameters:

folders an inx holder containing a list of folders to be searched for the specified folder.

messages an inx holder containing a complete list of a user's messages to be examined for gathering statistics.

name a managed string containing the name of the imap folder to be queried.

status a pointer to an imap folder status object to receive the folder's status information.

Returns:

1 on success or ≤ 0 on failure. 0: General failure. -1: The specified folder name was invalid. -2: The folder did not exist.

Definition at line 678 of file folders.c.

References `imap_folder_status_t::first`, `imap_folder_status_t::foldernum`, `imap_valid_folder_name()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `imap_folder_status_t::messages`, `meta_folder_t`, `meta_folders_by_name()`, `meta_message_t`, `mm_wipe()`, `imap_folder_status_t::recent`, `imap_folder_status_t::uidnext`, and `imap_folder_status_t::unseen`.

Referenced by `imap_examine()`, `imap_select()`, and `imap_status()`.

5.151.2.43 `imap_arguments_t* imap_get_ar_ar (imap_arguments_t * array, size_t element)`

Return the value of a specified object in an array as an imap arguments array.

Parameters:

array a pointer to an imap arguments array.

element the zero-based index of the object in the imap arguments array to be examined.

Returns:

NULL on failure, or a pointer to the specified object's value as an imap arguments array on success.

Definition at line 100 of file parse.c.

References `ar_length_get()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_type_ar()`, and `log_pedantic`.

Referenced by `imap_fetch_body()`, `imap_id()`, `imap_parse_dataitems()`, `imap_search_messages_inner()`, and `imap_status()`.

5.151.2.44 `uint32_t imap_get_flag (stringer_t * string)`

Definition at line 10 of file flags.c.

References `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `PLACER`, and `st_cmp_ci_eq()`.

Referenced by `imap_flag_parse()`.

5.151.2.45 `void* imap_get_ptr (imap_arguments_t * array, size_t element)`

Return the value of a specified object in an array as a generic pointer.

Parameters:

array a pointer to an imap arguments array.

element the zero-based index of the object in the imap arguments array to be examined.

Returns:

NULL on failure, or a pointer to the specified object's value on success.

Definition at line 46 of file parse.c.

References `ar_length_get()`, and `log_pedantic`.

Referenced by `imap_append()`, `imap_fetch_body()`, and `imap_store()`.

5.151.2.46 stringer_t* imap_get_st_ar (imap_arguments_t * array, size_t element)

Return the value of a specified object in an array as a managed string.

Parameters:

array a pointer to an imap arguments array.

element the zero-based index of the object in the imap arguments array to be examined.

Returns:

NULL on failure, or a pointer to the specified object's value as a managed string on success.

Definition at line 71 of file parse.c.

References ar_length_get(), IMAP_ARGUMENT_TYPE_ARRAY, imap_get_type_ar(), and log_pedantic.

Referenced by imap_append(), imap_copy(), imap_create(), imap_delete(), imap_examine(), imap_fetch(), imap_fetch_body(), imap_fetch_body_header(), imap_fetch_body_tag(), imap_flag_parse(), imap_id(), imap_list(), imap_login(), imap_lsub(), imap_parse_dataitems(), imap_rename(), imap_search_messages_inner(), imap_select(), imap_status(), imap_store(), and imap_subscribe().

5.151.2.47 int_t imap_get_type_ar (imap_arguments_t * array, size_t element)

TODO: The logic here is just plain ugly. Specifically the logic used to read from the network and advance the buffers. Get the type code for a specified object in an array.

Note:

Valid return values include IMAP_ARGUMENT_TYPE_ARRAY, IMAP_ARGUMENT_TYPE_ASTRING, IMAP_ARGUMENT_TYPE_QSTRING, IMAP_ARGUMENT_TYPE_NSTRING, and IMAP_ARGUMENT_TYPE_LITERAL.

Parameters:

array a pointer to an imap arguments array.

element the zero-based index of the object in the imap arguments array to be examined.

Returns:

IMAP_ARGUMENT_TYPE_EMPTY on failure, or the element type code of the specified object on success.

Definition at line 20 of file parse.c.

References ar_length_get(), IMAP_ARGUMENT_TYPE_EMPTY, and log_pedantic.

Referenced by imap_append(), imap_copy(), imap_create(), imap_delete(), imap_examine(), imap_fetch(), imap_fetch_body(), imap_fetch_body_header(), imap_flag_parse(), imap_get_ar_ar(), imap_get_st_ar(), imap_id(), imap_list(), imap_login(), imap_lsub(), imap_parse_dataitems(), imap_rename(), imap_search_messages_inner(), imap_select(), imap_status(), imap_store(), and imap_subscribe().

5.151.2.48 void imap_id (connection_t * con)

Definition at line 1782 of file imap.c.

References ar_length_get(), build_stamp(), build_version(), con_addr_presentation(), con_print(), count, IMAP_ARGUMENT_TYPE_ARRAY, imap_get_ar_ar(), imap_get_st_ar(), imap_get_type_ar(), log_info, MANAGEDBUF, st_append_opts(), st_char_get(), st_free(), st_length_get(), st_length_int(), st_sprint(), and time_print_local().

5.151.2.49 void imap_idle (connection_t * con)

Definition at line 1765 of file imap.c.

References con_print(), st_char_get(), and st_length_int().

5.151.2.50 void imap_init (connection_t * con)

The main IMAP entry point for all inbound client connections, as dispatched by the generic protocol handler (display banner greeting).

Parameters:

con a pointer to the connection object of the newly connected client.

Returns:

This function returns no value.

Definition at line 1853 of file imap.c.

References build_version(), con_print(), con_reverse_enqueue(), con_secure(), imap_requeue(), st_char_get(), st_length_get(), and st_length_int().

Referenced by protocol_enqueue().

5.151.2.51 void imap_invalid (connection_t * con)

Respond to an invalid IMAP command from a client.

Parameters:

con a pointer to the client connection that issued the bad command.

Returns:

This function returns no value.

Definition at line 63 of file imap.c.

References con_print(), st_char_get(), and st_length_int().

Referenced by imap_process(), and imap_starttls().

5.151.2.52 void imap_list (connection_t * con)

Definition at line 312 of file imap.c.

References ar_length_get(), con_print(), IMAP_ARGUMENT_TYPE_ARRAY, imap_folder_name_escaped(), imap_get_st_ar(), imap_get_type_ar(), imap_narrow_folders(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_reset(), inx_cursor_t, inx_cursor_value_next(), inx_free(), inx_t, log_pedantic, magma_folder_name(), meta_folder_t, meta_user_rlock(), meta_user_unlock(), NULLER, PLACER, st_char_get(), st_cmp_ci_eq(), st_free(), and st_length_int().

5.151.2.53 void imap_login (connection_t * con)

Attempt to perform a user login on an IMAP client connection.

Parameters:

con a pointer to the connection object of the remote session.

Returns:

This function returns no value.

Definition at line 106 of file imap.c.

References `ar_length_get()`, `auth_free()`, `AUTH_LOCK_ABUSE`, `AUTH_LOCK_ADMIN`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_USER`, `auth_login()`, `cache_decrement()`, `cache_increment()`, `con_addr_presentation()`, `con_addr_subnet()`, `con_print()`, `con_secure()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_st_ar()`, `imap_get_type_ar()`, `inx_count()`, `auth_t::keys`, `lock_get()`, `lock_release()`, `auth_t::locked`, `log_info`, `MANAGEDBUF`, `auth_t::master`, `meta_data_update_log()`, `meta_get()`, `META_GET_FOLDERS`, `META_GET_KEYS`, `META_GET_MESSAGES`, `meta_inx_remove()`, `META_PROTOCOL_IMAP`, `META_USER_ENCRYPT_DATA`, `meta_user_rlock()`, `META_USER_TLS`, `meta_user_unlock()`, `auth_t::salt`, `auth_t::seasoning`, `st_char_get()`, `st_dupe()`, `st_free()`, `st_length_int()`, `st_populated`, `st_quick()`, `auth_t::status`, `time_datestamp()`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, and `auth_t::verification`.

5.151.2.54 `void imap_logout (connection_t * con)`

Terminate an IMAP session gracefully with a "BYE" message and destroy the underlying connection.

Parameters:

con the IMAP connection to be terminated.

Returns:

This function returns no value.

Definition at line 79 of file `imap.c`.

References `con_destroy()`, `con_print()`, `con_status()`, `con_write_bl()`, `st_char_get()`, `st_length_int()`, and `status`.

Referenced by `imap_process()`, and `imap_requeue()`.

5.151.2.55 `void imap_lsub (connection_t * con)`

Definition at line 397 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_name_escaped()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_narrow_folders()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_reset()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `meta_folder_t`, `meta_user_rlock()`, `meta_user_unlock()`, `NULLER`, `PLACER`, `st_char_get()`, `st_cmp_ci_eq()`, `st_free()`, and `st_length_int()`.

5.151.2.56 `int_t imap_message_copier (connection_t * con, meta_message_t * message, uint64_t target, uint64_t * outnum)`

Definition at line 65 of file `messages.c`.

References `inx_append()`, `log_pedantic`, `M_TYPE_UINT64`, `mail_copy_message()`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_HIDDEN`, `MAIL_STATUS_RECENT`, `meta_message_dupe()`, `meta_message_t`, `meta_messages_update_sequences()`, `mm_free()`, `OBJECT_MESSAGES`, `serial_get()`, `serial_increment()`, `status`, `multi_t::type`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_copy()`.

5.151.2.57 `int_t imap_message_expunge (connection_t * con, meta_message_t * message)`

Definition at line 53 of file `messages.c`.

References `inx_delete()`, `M_TYPE_UINT64`, and `mail_remove_message()`.

Referenced by `imap_close()`, and `imap_expunge()`.

5.151.2.58 `inx_t* imap_narrow_folders (inx_t * folders, stringer_t * reference, stringer_t * mailbox)`

Definition at line 743 of file `folders.c`.

References `imap_folder_compare()`, `inx_alloc()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_insert()`, `inx_t`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_folder_t`, `meta_folders_name()`, `mm_dupe()`, `mm_free()`, `st_free()`, `st_length_get()`, `st_merge`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_list()`, and `imap_lsub()`.

5.151.2.59 `inx_t* imap_narrow_messages (inx_t * messages, uint64_t selected, stringer_t * range, int_t uid)`

Definition at line 1279 of file `fetch.c`.

References `inx_alloc()`, `inx_append()`, `inx_cleanup()`, `inx_count()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `log_error`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_message_t`, `number`, `pl_char_get()`, `pl_empty()`, `pl_null()`, `placer_t`, `st_char_get()`, `st_length_int()`, `tok_get_count_st()`, `tok_get_st()`, `multi_t::u64`, `uint64_conv_st()`, and `multi_t::val`.

Referenced by `imap_copy()`, `imap_fetch()`, and `imap_store()`.

5.151.2.60 `uint64_t imap_next_folder_order (inx_t * folders, uint64_t parent)`

Get the next order value for an imap folder.

Note:

The next order value is the current highest order value of any child folder in the specified directory, incremented by one.

Parameters:

folder an `inx` holder containing the collection of imap folders to be scanned.

parent the numerical id of the folder to be queried.

Returns:

0 on failure, or the next order value of the specified folder on success.

Definition at line 67 of file `folders.c`.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, and `meta_folder_t`.

Referenced by `imap_folder_create()`, and `imap_folder_rename()`.

5.151.2.61 `void imap_noop (connection_t * con)`

Definition at line 286 of file `imap.c`.

References `con_print()`, `imap_session_update()`, `st_char_get()`, and `st_length_int()`.

5.151.2.62 `stringer_t* imap_parse_address (stringer_t * address)`

[parse_address.c](#)

Definition at line 211 of file `parse_address.c`.

References `imap_build_array()`, `imap_parse_address_breaker()`, `imap_parse_address_part()`, `pl_empty()`, `placer_t`, `st_free()`, `st_merge`, and `imap_fetch_response_t::value`.

Referenced by `imap_fetch_envelope()`.

5.151.2.63 `placer_t imap_parse_address_breaker (stringer_t * address, uint32_t part)`

Definition at line 158 of file `parse_address.c`.

References `length`, `pl_init()`, `pl_null()`, `placer_t`, `st_char_get()`, `st_length_get()`, and `status`.

Referenced by `imap_parse_address()`.

5.151.2.64 `stringer_t* imap_parse_address_part (placer_t input)`

Definition at line 27 of file `parse_address.c`.

References `imap_build_array()`, `imap_parse_address_put()`, `length`, `pl_data_get()`, `pl_length_get()`, `placer_t`, `st_alloc()`, `st_char_get()`, `st_free()`, `st_length_get()`, `st_length_set()`, and `tok_get_st()`.

Referenced by `imap_parse_address()`.

5.151.2.65 `void imap_parse_address_put (stringer_t * buffer, chr_t c)`

LOW: Do we need an `imap_parse_address_group()` function?

Definition at line 12 of file `parse_address.c`.

References `st_avail_get()`, `st_char_get()`, `st_length_get()`, and `st_length_set()`.

Referenced by `imap_parse_address_part()`.

5.151.2.66 `int_t imap_parse_arguments (connection_t * con, chr_t ** start, size_t * length)`

Parse a chunk of input into an array of IMAP arguments.

See also:

[imap_parse_qstring\(\)](#), [imap_parse_literal\(\)](#), [imap_parse_array\(\)](#), [imap_parse_astring\(\)](#)

Note:

This function scans a string for an array of arguments, performing the following logic when a particular character is encountered: `:` Parse argument as quoted string with [imap_parse_qstring\(\)](#). `{` : Parse argument as literal string with [imap_parse_literal\(\)](#). `(` or `[` : Parse argument as array with [imap_parse_astring\(\)](#). `other` : Defaults to parsing argument as atomic string with `imap_parse_atomic()`. The supplied `start` and `length` pointers will be updated to reflect the input stream if the quoted string is parsed successfully.

Parameters:

start the address of a pointer to the start of the buffer to be parsed, that will be continually updated to point to the next argument in the sequence during the parsing loop.

length a pointer to a `size_t` variable that contains the length of the string to be parsed, that will be continually updated with the input stream position during the parsing loop.

Returns:

-1 on parsing error or 1 on success.

Definition at line 709 of file `parse.c`.

References `ar_append()`, `ar_free()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `IMAP_ARGUMENT_TYPE_ASTRING`, `IMAP_ARGUMENT_TYPE_LITERAL`, `IMAP_ARGUMENT_TYPE_QSTRING`, `imap_parse_array()`, `imap_parse_astring()`, `imap_parse_literal()`, and `imap_parse_qstring()`.

Referenced by `imap_command_parser()`.

5.151.2.67 `int_t imap_parse_array (int_t recursion, connection_t * con, imap_arguments_t ** array, chr_t ** start, size_t * length)`

Extract the contents of an array argument and advance the position of the parser stream.

Note:

This function expects as input a string beginning either with '(' or '['.

Parameters:

length a pointer to a `size_t` variable that contains the length of the string to be parsed, and that will be updated to reflect the length of the remainder of the input string that follows the parsed literal string.

recursion `d`

con the client IMAP connection passing the array as input to the server.

array -

start -

length -

Returns:

Definition at line 595 of file `parse.c`.

References `ar_append()`, `ar_free()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `IMAP_ARGUMENT_TYPE_LITERAL`, `IMAP_ARGUMENT_TYPE_NSTRING`, `IMAP_ARGUMENT_TYPE_QSTRING`, `IMAP_ARRAY_RECURSION_LIMIT`, `imap_parse_array()`, `imap_parse_literal()`, `imap_parse_nstring()`, `imap_parse_qstring()`, `log_pedantic`, and `type()`.

Referenced by `imap_parse_arguments()`, and `imap_parse_array()`.

5.151.2.68 int_t imap_parse_astring (stringer_t ** output, chr_t ** start, size_t * length)

Extract the contents of an atomic string and advance the position of the parser stream.

Note:

This function scans a string, expecting printable ASCII characters until it encounters a space, , , (, or [. If any other character is encountered before that point, an error will be returned. The supplied start and length pointers will be updated to reflect the input stream if the quoted string is parsed successfully.

Parameters:

output the address of a managed string that will receive a copy of the atomic string's contents on success, or NULL on failure.

start the address of a pointer to the start of the buffer to be parsed, that will also be updated to point to the next argument in the sequence on success.

length a pointer to a `size_t` variable that contains the length of the string to be parsed, and which will be updated to reflect the length of the remainder of the input string that follows the parsed atomic string.

Returns:

-1 on general error or if an invalid character was encountered, or 1 if the supplied atomic string was valid.

Definition at line 154 of file `parse.c`.

References `log_pedantic`, and `st_import()`.

Referenced by `imap_command_parser()`, and `imap_parse_arguments()`.

5.151.2.69 imap_fetch_dataitems_t* imap_parse_dataitems (imap_arguments_t * arguments)

Definition at line 44 of file `fetch.c`.

References `ar_append()`, `ar_length_get()`, `ARRAY_TYPE_ARRAY`, `ARRAY_TYPE_POINTER`, `imap_fetch_dataitems_t::body`, `imap_fetch_dataitems_t::bodystructure`, `imap_fetch_dataitems_t::envelope`, `imap_fetch_dataitems_t::flags`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_fetch_free_items()`, `imap_get_ar_ar()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_fetch_dataitems_t::internaldate`, `log_error`, `mm_alloc()`, `imap_fetch_dataitems_t::normal`, `imap_fetch_dataitems_t::normal_partial`, `number`, `imap_fetch_dataitems_t::peek`, `imap_fetch_dataitems_t::peek_partial`, `PLACER`, `imap_fetch_dataitems_t::rfc822`, `imap_fetch_dataitems_t::rfc822_header`, `imap_fetch_dataitems_t::rfc822_size`, `imap_fetch_dataitems_t::rfc822_text`, `st_cmp_ci_eq()`, `st_cmp_cs_starts()`, `type()`, and `imap_fetch_dataitems_t::uid`.

Referenced by `imap_fetch()`.

5.151.2.70 `int_t imap_parse_literal (connection_t * con, stringer_t ** output, chr_t ** start, size_t * length)`

Extract the contents of a literal string and advance the position of the parser stream.

Note:

This function expects as input a string beginning with '{' and followed by a numerical string, an optional '+', and a closing '}'. After reading in the numerical size parameter, it then attempts to read in that many bytes of input from the network stream.

Parameters:

con the client IMAP connection passing the literal string as input to the server.

output the address of a managed string that will receive a copy of the literal string's contents on success, or NULL on failure or if it is zero length.

start the address of a pointer to the start of the buffer to be parsed (beginning with '{'), that will also be updated to point to the next argument in the sequence on success.

length a pointer to a size_t variable that contains the length of the string to be parsed, and that will be updated to reflect the length of the remainder of the input string that follows the parsed literal string.

Returns:

-1 on general or parse error or if an enclosing pair of double quotes was not found, or 1 if the supplied quoted string was valid.

Definition at line 381 of file `parse.c`.

References `con_read()`, `con_read_line()`, `con_write_bl()`, `line_pl_st()`, `log_pedantic`, `mm_copy()`, `mm_move()`, `number`, `pl_empty()`, `pl_length_get()`, `pl_null()`, `st_alloc()`, `st_char_get()`, `st_free()`, `st_length_set()`, and `uint64_conv_bl()`.

Referenced by `imap_parse_arguments()`, and `imap_parse_array()`.

5.151.2.71 `int_t imap_parse_nstring (stringer_t ** output, chr_t ** start, size_t * length, chr_t type)`

Definition at line 197 of file `parse.c`.

References `log_error`, and `st_import()`.

Referenced by `imap_parse_array()`.

5.151.2.72 `int_t imap_parse_qstring (stringer_t ** output, chr_t ** start, size_t * length)`

Extract the contents of a quoted string and advance the position of the parser stream.

Note:

This function scans a string beginning with '"' for a terminating '"', returning an error if or is encountered first. It is also able to handle escaped characters. The supplied start and length pointers will be updated to reflect the input stream if the quoted string is parsed successfully.

Parameters:

- output** the address of a managed string that will receive a copy of the quoted string's contents on success, or NULL on failure or if it is zero length.
- start** the address of a pointer to the start of the buffer to be parsed (beginning with \), that will also be updated to point to the next argument in the sequence on success.
- length** a pointer to a size_t variable that contains the length of the string to be parsed, and which will be updated to reflect the length of the remainder of the input string that follows the parsed quoted string.

Returns:

- 1 on general error or if an enclosing pair of double quotes was not found, or 1 if the supplied quoted string was valid.

Definition at line 251 of file parse.c.

References log_pedantic, st_alloc(), st_char_get(), and st_length_set().

Referenced by imap_parse_arguments(), and imap_parse_array().

5.151.2.73 void imap_process (connection_t * con)

Perform client command processing on an established imap session.

Note:

- This function will read the next line of user input, parse the command, and then attempt to execute it with the appropriate handler.

Parameters:

- con** a pointer to the connection object underlying the imap session.

Returns:

- This function returns no value.

Definition at line 61 of file commands.c.

References command, command_t, con_print(), con_read_line(), con_write_bl(), enqueue(), imap_command_parser(), imap_commands, imap_compare(), imap_invalid(), imap_logout(), imap_process(), imap_requeue(), pl_empty(), requeue(), st_char_get(), st_length_get(), and st_length_int().

Referenced by imap_process(), and imap_requeue().

5.151.2.74 stringer_t* imap_range_build (size_t length, uint64_t * numbers)

[range.c](#)

Definition at line 10 of file range.c.

References count, log_pedantic, st_free(), and st_merge.

Referenced by imap_copy().

5.151.2.75 void imap_rename (connection_t * con)

Definition at line 590 of file imap.c.

References ar_length_get(), con_print(), FOLDER_LENGTH_LIMIT, IMAP_ARGUMENT_TYPE_ARRAY, IMAP_FOLDER_RECURSION_LMIIT, imap_folder_rename(), imap_get_st_ar(), imap_get_type_ar(), meta_user_unlock(), meta_user_wlock(), OBJECT_FOLDERS, serial_get(), serial_increment(), st_char_get(), and st_length_int().

5.151.2.76 void imap_requeue (connection_t * con)

Requeue an imap connection for processing, or log it out if there was an error or excess of protocol violations.

Parameters:

con a pointer to the connection object of the imap session.

Returns:

This function returns no value.

Definition at line 43 of file commands.c.

References con_status(), enqueue(), imap_logout(), imap_process(), and status.

Referenced by imap_init(), and imap_process().

5.151.2.77 void imap_search (connection_t * con)

Definition at line 1712 of file imap.c.

References ar_length_get(), con_print(), con_write_st(), HEAP, imap_search_messages(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_free(), inx_t, JOINTED, MANAGED_T, MANAGEDBUF, meta_message_t, PLACER, st_append_opts(), st_aprint_opts(), st_char_get(), st_cleanup, st_length_int(), st_populated, st_sprint(), and status.

5.151.2.78 int_t imap_search_flag (uint32_t status, uint32_t flag, int_t has)

search.c

Definition at line 266 of file search.c.

Referenced by imap_search_messages_inner().

5.151.2.79 inx_t* imap_search_messages (connection_t * con)

LOW: Is a read lock necessary now that were using index reference counters and thread safe iteration cursors?

Definition at line 563 of file search.c.

References count, imap_search_messages_inner(), inx_alloc(), inx_append(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_key_active(), inx_cursor_t, inx_cursor_value_next(), inx_find(), inx_t, M_INX_LINKED, M_TYPE_UINT64, mail_destroy(), mail_destroy_header(), meta_message_dupe(), meta_message_free(), meta_message_t, meta_user_rlock(), meta_user_unlock(), status, multi_t::u64, and multi_t::val.

Referenced by imap_search().

5.151.2.80 int_t imap_search_messages_body (connection_t * con, meta_user_t * user, mail_message_t ** data, meta_message_t * active, stringer_t * value)

Definition at line 197 of file search.c.

References mail_load_message(), mail_mime_update(), st_free(), and st_search_ci().

Referenced by imap_search_messages_inner().

5.151.2.81 int_t imap_search_messages_date (connection_t * con, meta_user_t * user, mail_message_t ** data, stringer_t ** header, meta_message_t * active, stringer_t * date, int_t internal, int_t expected)

Definition at line 82 of file search.c.

References `imap_search_messages_date_compare()`, `mail_header_fetch_cleaned()`, `mail_load_header()`, `ns_length_get()`, `pl_empty()`, `pl_init()`, `pl_null()`, `PLACER`, `placer_t`, `st_char_get()`, `st_free()`, `st_import()`, `st_length_get()`, `st_merge`, `tok_get_count_st()`, and `tok_get_st()`.

Referenced by `imap_search_messages_inner()`.

5.151.2.82 `int_t imap_search_messages_date_compare (stringer_t * one, stringer_t * two)`

Definition at line 12 of file `search.c`.

References `MONTH_LOOKUP`, `PLACER`, `placer_t`, `st_cmp_ci_eq()`, `tok_get_count_st()`, `tok_get_st()`, and `uint32_conv_st()`.

Referenced by `imap_search_messages_date()`.

5.151.2.83 `int_t imap_search_messages_header (connection_t * con, meta_user_t * user, mail_message_t ** data, stringer_t ** header, meta_message_t * active, stringer_t * field, stringer_t * value)`

Definition at line 161 of file `search.c`.

References `mail_header_fetch_all()`, `mail_load_header()`, `pl_empty()`, `pl_init()`, `pl_null()`, `placer_t`, `st_char_get()`, `st_free()`, `st_length_get()`, and `st_search_ci()`.

Referenced by `imap_search_messages_inner()`.

5.151.2.84 `int_t imap_search_messages_inner (connection_t * con, meta_user_t * user, mail_message_t ** message, stringer_t ** header, meta_message_t * current, imap_arguments_t * array, unsigned recursion)`

Definition at line 347 of file `search.c`.

References `ar_length_get()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_ar_ar()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_search_flag()`, `imap_search_messages_body()`, `imap_search_messages_date()`, `imap_search_messages_header()`, `imap_search_messages_inner()`, `imap_search_messages_range()`, `imap_search_messages_size()`, `imap_search_messages_text()`, `IMAP_SEARCH_RECURSION_LIMIT`, `imap_valid_sequence()`, `log_pedantic`, `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `number`, `PLACER`, and `st_cmp_ci_eq()`.

Referenced by `imap_search_messages()`, and `imap_search_messages_inner()`.

5.151.2.85 `int_t imap_search_messages_range (meta_message_t * active, stringer_t * range, int_t uid)`

Definition at line 276 of file `search.c`.

References `number`, `pl_char_get()`, `pl_empty()`, `pl_null()`, `placer_t`, `tok_get_count_st()`, `tok_get_st()`, and `uint64_conv_st()`.

Referenced by `imap_search_messages_inner()`.

5.151.2.86 `int_t imap_search_messages_size (meta_message_t * active, stringer_t * value, int_t expected)`

Definition at line 245 of file `search.c`.

References `uint64_conv_st()`.

Referenced by `imap_search_messages_inner()`.

5.151.2.87 `int_t imap_search_messages_text (connection_t * con, meta_user_t * user, mail_message_t ** data, meta_message_t * active, stringer_t * value)`

Definition at line 221 of file `search.c`.

References `mail_load_message()`, `mail_mime_update()`, `st_free()`, and `st_search_ci()`.

Referenced by `imap_search_messages_inner()`.

5.151.2.88 void imap_select (connection_t * con)

Definition at line 873 of file imap.c.

References ar_length_get(), con_print(), imap_folder_status_t::first, imap_folder_status_t::foldernum, IMAP_ARGUMENT_TYPE_ARRAY, imap_folder_status(), imap_get_st_ar(), imap_get_type_ar(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_RECENT, imap_folder_status_t::messages, meta_data_flags_remove(), meta_message_t, meta_user_unlock(), meta_user_wlock(), imap_folder_status_t::recent, st_char_get(), st_length_int(), status, and imap_folder_status_t::uidnext.

5.151.2.89 void imap_session_destroy (connection_t * con)

sessions.c

Definition at line 93 of file sessions.c.

References ar_free(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), mail_cache_reset(), MAIL_STATUS_RECENT, meta_inx_remove(), meta_message_t, META_PROTOCOL_IMAP, meta_user_unlock(), meta_user_wlock(), and st_cleanup.

Referenced by con_destroy().

5.151.2.90 int_t imap_session_update (connection_t * con)

Returns 0 if the selected folder wasn't modified, or 1 if things changed and the updated status should be sent to the client, and a -1 is used to indicate the update check encountered a problem and should be retried later.

Definition at line 14 of file sessions.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_RECENT, META_LOCKED, meta_message_t, meta_messages_update(), meta_update_folders(), meta_update_user(), meta_user_unlock(), meta_user_wlock(), OBJECT_FOLDERS, OBJECT_MESSAGES, OBJECT_USER, and serial_get().

Referenced by imap_check(), imap_expunge(), imap_noop(), and imap_store().

5.151.2.91 void imap_sort (void)

Sort the IMAP command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 33 of file commands.c.

References command_t, imap_commands, and imap_compare().

Referenced by protocol_init().

5.151.2.92 void imap_starttls (connection_t * con)

Create a secure connection for an IMAP session. TODO: Review error messages and update them with the appropriate response code. LOW: When should we check the serial number to see if the local data is stale and needs to be refreshed? LOW: Add function descriptions to all of the different IMAP commands.

Note:

RFC 2595 / section 3.1 specifies that STARTTLS is only available in a non-authenticated state.

Parameters:

con the connection on top of which the TLS session will be established.

Returns:

This function returns no value.

Definition at line 23 of file `imap.c`.

References `con_print()`, `con_secure()`, `imap_invalid()`, `log_pedantic`, `M_SSL_BIO_NOCLOSE`, `pl_null()`, `st_char_get()`, `st_length_get()`, `st_length_set()`, `stats_increment_by_name()`, and `tls_server_alloc()`.

5.151.2.93 void imap_status (connection_t * con)

Definition at line 656 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `imap_folder_status_t::foldernum`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_status()`, `imap_get_ar_ar()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_folder_status_t::messages`, `meta_user_rlock()`, `meta_user_unlock()`, `NULLER`, `number`, `PLACER`, `imap_folder_status_t::recent`, `st_append_opts()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_free()`, `st_length_get()`, `st_length_int()`, `status`, `imap_folder_status_t::uidnext`, `imap_folder_status_t::unseen`, and `values`.

5.151.2.94 void imap_store (connection_t * con)

LOW: Shouldn't we be checking for stale status info so the update doesn't make decisions based on incorrect status data? On the other hand the actual IMAP logic is passed all the way through to the DB so even if the server ends up with incorrect status information, the database should remain accurate.

Definition at line 952 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_duplicate_messages()`, `imap_flag_action()`, `imap_flag_parse()`, `IMAP_FLAG_SILENT`, `imap_get_ptr()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_narrow_messages()`, `imap_session_update()`, `imap_update_flags()`, `imap_valid_sequence()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `meta_message_t`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_MESSAGES`, `serial_get()`, `serial_increment()`, `st_char_get()`, and `st_length_int()`.

5.151.2.95 void imap_subscribe (connection_t * con)

Definition at line 743 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `FOLDER_LENGTH_LIMIT`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_create()`, `IMAP_FOLDER_RECURSION_LIMIT`, `imap_get_st_ar()`, `imap_get_type_ar()`, `meta_folder_t`, `meta_folders_by_name()`, `meta_user_unlock()`, `meta_user_wlock()`, `st_char_get()`, and `st_length_int()`.

5.151.2.96 void imap_unsubscribe (connection_t * con)

Definition at line 792 of file `imap.c`.

References `con_print()`, `st_char_get()`, and `st_length_int()`.

5.151.2.97 void imap_update_flags (meta_user_t * user, inx_t * messages, uint64_t foldernum, int_t action, uint32_t flags)

Definition at line 112 of file `flags.c`.

References `IMAP_FLAG_ADD`, `IMAP_FLAG_REMOVE`, `IMAP_FLAG_REPLACE`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_error`, `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_EMPTY`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_flags_replace()`, and `meta_message_t`.

Referenced by `imap_store()`.

5.151.2.98 bool_t imap_valid_folder_name (stringer_t * *name*)

Check to see if a name is a valid imap folder name.

Note:

The following naming checks are enforced: 1. The name can't be empty or begin with a period. 2. It cannot contain a non-printable character. 3. Trailing periods will be removed. 4. The folder name cannot contain consecutive periods.

Parameters:

name a managed string containing the imap folder name to be validated.

Returns:

true on success or false on failure.

Definition at line 157 of file folders.c.

References log_pedantic, st_empty_out(), st_length_get(), and st_length_set().

Referenced by contact_folder_create(), contact_folder_rename(), imap_folder_create(), imap_folder_remove(), imap_folder_rename(), and imap_folder_status().

5.151.2.99 int_t imap_valid_sequence (stringer_t * *range*)

Definition at line 11 of file fetch.c.

References length, and st_empty_out().

Referenced by imap_copy(), imap_fetch(), imap_search_messages_inner(), and imap_store().

5.152 src/network/listeners.c File Reference

```
#include "magma.h"
```

Functions

- void [net_accept](#) ([server_t](#) *server)
- void [net_listen](#) (void)
- [bool_t](#) [net_init](#) ([server_t](#) *server)
[listeners.c](#)
- void [net_trigger](#) ([bool_t](#) verbose)
Triggers a connection for each server instance, allowing the the listeners to shutdown cleanly. And then purges any remaining sockets.
- void [net_shutdown](#) ([server_t](#) *server)
Close the listening socket associated with a server.

5.152.1 Function Documentation

5.152.1.1 void net_accept (server_t * server)

Definition at line 10 of file listeners.c.

References [server_t::network](#), [protocol_process\(\)](#), [server_t::sockd](#), [status](#), [thread_start\(\)](#), and [thread_stop\(\)](#).

Referenced by [net_listen\(\)](#).

5.152.1.2 bool_t net_init (server_t * server)

[listeners.c](#)

Definition at line 58 of file listeners.c.

References [server_t::ipv6](#), [server_t::listen_queue](#), [log_critical](#), [magma](#), [mm_wipe\(\)](#), [net_set_buffer_length\(\)](#), [net_set_reuseable_address\(\)](#), [server_t::network](#), [magma_t::network_buffer](#), [server_t::port](#), [server_t::sockd](#), and [magma_t::system](#).

Referenced by [servers_network_start\(\)](#).

5.152.1.3 void net_listen (void)

Definition at line 33 of file listeners.c.

References [server_t::enabled](#), [magma](#), [MAGMA_SERVER_INSTANCES](#), [mm_free\(\)](#), [mm_wipe\(\)](#), [net_accept\(\)](#), [server_t::network](#), [magma_t::servers](#), [server_t::sockd](#), [thread_alloc\(\)](#), and [thread_join\(\)](#).

Referenced by [main\(\)](#).

5.152.1.4 void net_shutdown (server_t * server)

Close the listening socket associated with a server.

Returns:

This function returns no value.

Definition at line 216 of file listeners.c.

References `server_t::network`, and `server_t::sockd`.

Referenced by `servers_network_stop()`.

5.152.1.5 `void net_trigger (bool_t verbose)`

Triggers a connection for each server instance, allowing the the listeners to shutdown cleanly. And then purges any remaining sockets.

LOW: This only compares the port number for the sockets. We should also ensure the socket is owned by `magmad`, and/or that the server used `INADDR_ANY` or `IN6ADDR_ANY_INIT`, otherwise the logic below will close sockets that could be owned by other processes on the system that are using the same port number, while bound to a different IP interface.

Definition at line 129 of file listeners.c.

References `client_close()`, `client_connect()`, `client_t`, `server_t::enabled`, `ip_t::family`, `ip_t::ip4`, `ip_t::ip6`, `ip_presentation()`, `log_critical`, `log_info`, `magma`, `MAGMA_SERVER_INSTANCES`, `MEMORYBUF`, `mm_copy()`, `mm_wipe()`, `server_t::network`, `PLACER`, `server_t::port`, `queue_signal()`, `magma_t::servers`, `servers_get_count_using_port()`, `server_t::sockd`, and `st_char_get()`.

Referenced by `signal_shutdown()`.

5.153 src/network/meta.h File Reference

Enumerations

- enum `META_GET` {
`META_GET_NONE` = 0, `META_GET_KEYS` = 1, `META_GET_ALIASES` = 2, `META_GET_MESSAGES` = 43,
`META_GET_FOLDERS` = 8, `META_GET_CONTACTS` = 16 }
- enum `META_LOCK_STATUS` { `META_NEED_LOCK` = 0, `META_LOCKED` = 1 }
- enum `META_CHECKPOINT` { `META_CHECKPOINT_MESSAGES` = 0, `META_CHECKPOINT_FOLDERS` = 1, `META_CHECKPOINT_USER` = 2 }
- enum `META_USER_FLAGS` {
`META_USER_NONE` = 0, `META_USER_TLS` = 1, `META_USER_ADVERTISING` = 2, `META_USER_OVERQUOTA` = 4,
`META_USER_ENCRYPT_DATA` = 8 }
- enum `META_PROTOCOL` {
`META_PROTOCOL_NONE` = 0, `META_PROTOCOL_SMTP` = 1, `META_PROTOCOL_DMTP` = 2, `META_PROTOCOL_POP` = 4,
`META_PROTOCOL_IMAP` = 8, `META_PROTOCOL_DMAP` = 16, `META_PROTOCOL_WEB` = 32, `META_PROTOCOL_JSON` = 64,
`META_PROTOCOL_GENERIC` = 128 }
- enum { `CREDENTIAL_MAIL` = 0, `CREDENTIAL_AUTH` = 1 }

Functions

- struct `__attribute__` ((packed))

Variables

- `meta_alert_t`
- `meta_alias_t`
- `meta_message_t`
- `meta_folder_t`
- `meta_user_t`

5.153.1 Enumeration Type Documentation

5.153.1.1 anonymous enum

Enumerator:

CREDENTIAL_MAIL
CREDENTIAL_AUTH

Definition at line 60 of file meta.h.

5.153.1.2 enum META_CHECKPOINT

Enumerator:

META_CHECKPOINT_MESSAGES
META_CHECKPOINT_FOLDERS
META_CHECKPOINT_USER

Definition at line 25 of file meta.h.

5.153.1.3 enum META_GET

Enumerator:

META_GET_NONE
META_GET_KEYS
META_GET_ALIASES
META_GET_MESSAGES
META_GET_FOLDERS
META_GET_CONTACTS

Definition at line 11 of file meta.h.

5.153.1.4 enum META_LOCK_STATUS

Enumerator:

META_NEED_LOCK
META_LOCKED

Definition at line 20 of file meta.h.

5.153.1.5 enum META_PROTOCOL

Enumerator:

META_PROTOCOL_NONE
META_PROTOCOL_SMTP
META_PROTOCOL_DMTP
META_PROTOCOL_POP
META_PROTOCOL_IMAP
META_PROTOCOL_DMAP
META_PROTOCOL_WEB
META_PROTOCOL_JSON
META_PROTOCOL_GENERIC

Definition at line 39 of file meta.h.

5.153.1.6 enum META_USER_FLAGS

Enumerator:

META_USER_NONE
META_USER_TLS
META_USER_ADVERTISING
META_USER_OVERQUOTA
META_USER_ENCRYPT_DATA

Definition at line 31 of file meta.h.

5.153.2 Function Documentation

5.153.2.1 struct __attribute__((packed)) [read]

Definition at line 95 of file meta.h.

References `inx_t`, `lock`, `__attribute__::user`, `__attribute__::username`, and `__attribute__::usernum`.

5.153.3 Variable Documentation

5.153.3.1 meta_alert_t

Definition at line 68 of file meta.h.

Referenced by `alert_alloc()`, `meta_data_fetch_alerts()`, and `portal_endpoint_alert_list()`.

5.153.3.2 meta_alias_t

Definition at line 78 of file meta.h.

Referenced by `alias_alloc()`, `meta_data_fetch_mailbox_aliases()`, and `portal_endpoint_aliases()`.

5.153.3.3 meta_folder_t

Definition at line 92 of file meta.h.

Referenced by `imap_append()`, `imap_copy()`, `imap_folder_create()`, `imap_folder_remove()`, `imap_folder_rename()`, `imap_folder_status()`, `imap_list()`, `imap_lsub()`, `imap_narrow_folders()`, `imap_next_folder_order()`, `imap_subscribe()`, `meta_data_fetch_folders()`, `meta_folders_by_name()`, `meta_folders_by_number()`, `meta_folders_children()`, `meta_messages_update_sequences()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_rename()`, `portal_folder_mail_add()`, and `portal_folder_mail_remove()`.

5.153.3.4 meta_message_t

Definition at line 86 of file meta.h.

Referenced by `imap_append()`, `imap_append_message()`, `imap_close()`, `imap_copy()`, `imap_duplicate_messages()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_folder_remove()`, `imap_folder_status()`, `imap_message_copier()`, `imap_narrow_messages()`, `imap_search()`, `imap_search_messages()`, `imap_select()`, `imap_session_destroy()`, `imap_session_update()`, `imap_store()`, `imap_update_flags()`, `meta_data_fetch_messages()`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_flags_replace()`, `meta_folders_stats_tags()`, `meta_message_by_number()`, `meta_message_dupe()`, `meta_messages_copier()`, `meta_messages_update_sequences()`, `pop_dele()`, `pop_get_last()`, `pop_get_message()`, `pop_list()`, `pop_retr()`, `pop_session_destroy()`, `pop_session_reset()`, `pop_top()`, `pop_total_messages()`, `pop_total_size()`, `pop_uidl()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, and `portal_endpoint_messages_tag()`.

5.153.3.5 meta_user_t

Definition at line 124 of file meta.h.

Referenced by `__attribute__()`, `api_endpoint_auth()`, `meta_alloc()`, `meta_get()`, `meta_inx_find()`, `meta_inx_remove()`, `obj_cache_prune()`, and `portal_endpoint_auth()`.

5.154 src/objects/meta/meta.h File Reference

Data Structures

- struct [key_pair_t](#)
- struct [meta_stats_tag_t](#)

Functions

- [meta_alert_t * alert_alloc](#) (uint64_t alertnum, [stringer_t](#) *type, [stringer_t](#) *message, uint64_t created)
alerts.c
- [meta_stats_tag_t * meta_folder_stats_tag_alloc](#) ([stringer_t](#) *tag)
folders.c
- [meta_folder_t * meta_folders_by_name](#) ([inx_t](#) *folders, [stringer_t](#) *name)
Get a folder by its fully qualified name.
- [meta_folder_t * meta_folders_by_number](#) ([inx_t](#) *folders, uint64_t number)
Get a folder by number.
- [int_t meta_folders_children](#) ([inx_t](#) *folders, uint64_t number)
Get the number of direct child folders of a specified parent folder.
- [stringer_t * meta_folders_name](#) ([inx_t](#) *list, [meta_folder_t](#) *folder)
Get a folder's fully qualified name.
- [inx_t * meta_folders_stats_tags](#) ([inx_t](#) *messages, uint64_t folder)
Create a collection of meta tag stats for a set of messages.
- [bool_t meta_user_serial_check](#) ([meta_user_t](#) *user, uint64_t object)
serials.c
- [uint64_t meta_user_serial_get](#) ([meta_user_t](#) *user, uint64_t object)
Get an object serial number for a given meta object.
- [void meta_user_serial_set](#) ([meta_user_t](#) *user, uint64_t object, uint64_t serial)
Set an object serial number for a given meta user structure.
- [int_t meta_update_aliases](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
updaters.c
- [int_t meta_update_contacts](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Update a user's contacts if necessary.
- [int_t meta_update_folders](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Update a user's message folders if necessary.
- [int_t meta_update_keys](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Fetches the user signet and private key. Relies on the mail realm to decrypt the values.
- [int_t meta_update_message_folders](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)

Refresh a user's message folders if stale, or retrieve them from the database if they are not in memory.

- [int_t meta_update_realms](#) ([meta_user_t](#) *user, [stringer_t](#) *salt, [stringer_t](#) *master, [META_LOCK_STATUS](#) locked)
Fetches the user realm keys and extracts the different components.
- [int_t meta_update_user](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Build a user's meta information from specified data parameters.
- void [meta_user_ref_add](#) ([meta_user_t](#) *user, [META_PROTOCOL](#) protocol)
references.c
- void [meta_user_ref_dec](#) ([meta_user_t](#) *user, [META_PROTOCOL](#) protocol)
Decrement a user's reference counter for the specified protocol and update the activity timestamp.
- [uint64_t meta_user_ref_protocol_total](#) ([meta_user_t](#) *user, [META_PROTOCOL](#) protocol)
Get the total reference count for the user object over the specified protocol.
- [time_t meta_user_ref_stamp](#) ([meta_user_t](#) *user)
Get the activity timestamp for a meta user object.
- [uint64_t meta_user_ref_total](#) ([meta_user_t](#) *user)
Get the total reference count for a user object, over all of the supported protocols.
- [meta_user_t](#) * [meta_alloc](#) (void)
meta.c
- void [meta_free](#) ([meta_user_t](#) *user)
Free a meta user object.
- [int_t meta_get](#) ([uint64_t](#) usernum, [stringer_t](#) *username, [stringer_t](#) *salt, [stringer_t](#) *master, [stringer_t](#) *verification, [META_PROTOCOL](#) protocol, [META_GET](#) get, [meta_user_t](#) **output)
Lookup user and return their meta user object.
- [bool_t meta_data_acknowledge_alert](#) ([uint64_t](#) alertnum, [uint64_t](#) usernum, [uint32_t](#) transaction)
datatier.c
- [uint64_t meta_data_delete_folder](#) ([uint64_t](#) usernum, [uint64_t](#) foldernum)
Delete a message folder from the database.
- [int_t meta_data_delete_tag](#) ([meta_message_t](#) *message, [stringer_t](#) *tag)
Remove a tag from a message in the database.
- [inx_t](#) * [meta_data_fetch_alerts](#) ([uint64_t](#) usernum)
Get all unacknowledged alert messages for a user.
- [inx_t](#) * [meta_data_fetch_all_tags](#) ([uint64_t](#) usernum)
Fetch all the tags attached to messages of a specified user from the database.
- [bool_t meta_data_fetch_folders](#) ([meta_user_t](#) *user)
Retrieve all of a user's message folders from the database.
- [int_t meta_data_fetch_keys](#) ([meta_user_t](#) *user, [key_pair_t](#) *output, [int64_t](#) transaction)
Retrieve the encryption keys for a user account.

- [int_t meta_data_fetch_mailbox_aliases](#) ([meta_user_t](#) *user)
Get all the mailbox aliases for a specified user.
- [int_t meta_data_fetch_shard](#) (uint64_t usernum, uint16_t serial, [stringer_t](#) *label, [stringer_t](#) *output, [uint_t](#) *rotated, int64_t transaction)
Retrieve a shard value for user account.
- [int_t meta_data_fetch_user](#) ([meta_user_t](#) *user)
Build a meta user object by username, hashed password, and hashed key storage password.
- [bool_t meta_data_flags_add](#) ([inx_t](#) *messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)
Add the specified flags mask to a collection of mail messages.
- [bool_t meta_data_flags_remove](#) ([inx_t](#) *messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)
Remove the specified flags mask from a collection of mail messages.
- [bool_t meta_data_flags_replace](#) ([inx_t](#) *messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)
Remove all user (non-system) flags from a collection of mail messages, and set the specified flags mask for them.
- [uint64_t meta_data_insert_folder](#) (uint64_t usernum, [stringer_t](#) *name, uint64_t parent, uint32_t order)
Insert a new mail folder into the database.
- [int_t meta_data_insert_keys](#) (uint64_t usernum, [stringer_t](#) *username, [key_pair_t](#) *input, int64_t transaction)
Store the encryption keys for a user account.
- [int_t meta_data_insert_shard](#) (uint64_t usernum, uint16_t serial, [stringer_t](#) *label, [stringer_t](#) *shard, int64_t transaction)
Store the user realm shard value.
- [int_t meta_data_insert_tag](#) ([meta_message_t](#) *message, [stringer_t](#) *tag)
Insert a tag for a message into the database.
- [int_t meta_data_truncate_tags](#) ([meta_message_t](#) *message)
Remove all tags associated with a message in the database.
- [uint64_t meta_data_update_folder_name](#) (uint64_t usernum, uint64_t foldernum, [stringer_t](#) *name, uint64_t parent, uint32_t order)
Update the record for a message folder in the database.
- void [meta_data_update_lock](#) (uint64_t usernum, uint8_t lock)
- void [meta_data_update_log](#) ([meta_user_t](#) *user, [META_PROTOCOL](#) prot)
Update the per-user entry in the Log table for the specified protocol.
- void [meta_user_rlock](#) ([meta_user_t](#) *user)
locking.c
- void [meta_user_unlock](#) ([meta_user_t](#) *user)
Release the lock for a meta user object.
- void [meta_user_wlock](#) ([meta_user_t](#) *user)
Acquire a write lock for a meta user object.
- [meta_user_t](#) * [meta_inx_find](#) (uint64_t usernum, [META_PROTOCOL](#) protocol)
indexes.c

- void [meta_inx_remove](#) (uint64_t usernum, [META_PROTOCOL](#) protocol)
Lock a user's object in the cache and decrement their reference counter.
- [int_t meta_crypto_keys_create](#) (uint64_t usernum, [stringer_t](#) *username, [stringer_t](#) *realm, int64_t transaction)
crypto.c
- [meta_alias_t * alias_alloc](#) (uint64_t aliasnum, [stringer_t](#) *address, [stringer_t](#) *display, [int_t](#) selected, uint64_t created)
alias.c

5.154.1 Function Documentation

5.154.1.1 [meta_alert_t* alert_alloc](#) (uint64_t *alertnum*, [stringer_t](#) * *type*, [stringer_t](#) * *message*, uint64_t *created*)

[alerts.c](#) [alerts.c](#)

Parameters:

- alertnum*** the numerical id of the user alert.
- type*** a pointer to a managed string containing the type of the alert (e.g. "warning" or "alert").
- message*** a pointer to a managed string containing the alert message.
- created*** the UTC timecode for when the alert message was created.

Returns:

NULL on error or a pointer to the newly initialized user alert message object on success.

Definition at line 18 of file [alerts.c](#).

References [align\(\)](#), [FOREIGNDATA](#), [JOINTED](#), [log_pedantic](#), [meta_alert_t](#), [mm_alloc\(\)](#), [mm_copy\(\)](#), [PLACER_T](#), [placer_t](#), [st_data_get\(\)](#), [st_length_get\(\)](#), and [STACK](#).

Referenced by [meta_data_fetch_alerts\(\)](#).

5.154.1.2 [meta_alias_t* alias_alloc](#) (uint64_t *aliasnum*, [stringer_t](#) * *address*, [stringer_t](#) * *display*, [int_t](#) *selected*, uint64_t *created*)

[alias.c](#) [alias.c](#)

Parameters:

- aliasnum*** the numerical identifier of the user alias.
- address*** a managed string containing the email address associated with the alias.
- display*** a managed string containing the display name associated with the alias.
- selected*** a flag specifying whether this alias is the default selection for the user.
- created*** a UNIX timestamp for the creation of this alias.

Returns:

NULL on failure, or a pointer to the newly initialized user alias object on success.

Definition at line 19 of file [alias.c](#).

References [align\(\)](#), [FOREIGNDATA](#), [JOINTED](#), [log_pedantic](#), [meta_alias_t](#), [mm_alloc\(\)](#), [mm_copy\(\)](#), [PLACER_T](#), [placer_t](#), [st_data_get\(\)](#), [st_length_get\(\)](#), and [STACK](#).

Referenced by [meta_data_fetch_mailbox_aliases\(\)](#).

5.154.1.3 meta_user_t* meta_alloc (void)

meta.c meta.c

Returns:

NULL on failure, or a pointer to the newly allocated meta user object on success.

Definition at line 47 of file meta.c.

References log_pedantic, meta_user_t, mm_alloc(), mm_free(), mm_wipe(), mutex_init(), rwlock_attr_destroy(), rwlock_attr_init(), rwlock_attr_setkind(), rwlock_destroy(), and rwlock_init().

Referenced by meta_inx_find().

5.154.1.4 int_t meta_crypto_keys_create (uint64_t usernum, stringer_t * username, stringer_t * realm, int64_t transaction)

crypto.c crypto.c

Parameters:

user

master

Returns:

-1 if an error occurs, 0 if successful, and 1 if the insertion fails because a key already exists.

Definition at line 18 of file crypto.c.

References BINARY, log_info, log_pedantic, meta_data_insert_keys(), NONE, org_key, prime_cleanup(), prime_free(), prime_get(), prime_key_encrypt(), prime_key_generate(), prime_request_generate(), prime_request_sign(), prime_t, PRIME_USER_KEY, key_pair_t::private, key_pair_t::public, st_char_get(), st_cleanup, st_free(), and st_length_int().

Referenced by meta_update_keys(), and register_data_insert_user().

5.154.1.5 bool_t meta_data_acknowledge_alert (uint64_t alertnum, uint64_t usernum, uint32_t transaction)

datatier.c datatier.c

Note:

If the table is not updated immediately, another check is made to see if the alert is still pending. If so, false is returned.

Parameters:

alertnum the numerical id of the alert message to be acknowledged.

usernum the numerical id of the user to whom the alert message belongs.

transaction the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

true if the alert was acknowledged successfully, or false on failure.

Definition at line 584 of file datatier.c.

References log_pedantic, mm_wipe(), res_field_uint64(), res_row_next(), res_table_free(), stmt_exec_affected_conn(), and stmt_get_result_conn().

Referenced by portal_endpoint_alert_acknowledge().

5.154.1.6 uint64_t meta_data_delete_folder (uint64_t *usernum*, uint64_t *foldernum*)

Delete a message folder from the database.

Parameters:

usernum the numerical id of the user that owns the folder to be deleted.

foldernum the folder id of the message folder to be deleted.

Returns:

1 if the message folder was successfully, or <= 0 if there was an error.

Definition at line 932 of file `datatier.c`.

References `M_FOLDER_MESSAGES`, `mm_wipe()`, `stmt_exec_affected()`, and `type()`.

Referenced by `imap_folder_remove()`.

5.154.1.7 int_t meta_data_delete_tag (meta_message_t * *message*, stringer_t * *tag*)

Remove a tag from a message in the database.

Parameters:

message a pointer to the meta message object of the message to have the tag stripped.

tag a managed string containing the name of the tag to be deleted.

Returns:

0 on success or -1 on failure.

Definition at line 1137 of file `datatier.c`.

References `mm_wipe()`, `st_char_get()`, `st_length_get()`, and `stmt_exec_affected()`.

Referenced by `portal_endpoint_messages_tag()`.

5.154.1.8 inx_t* meta_data_fetch_alerts (uint64_t *usernum*)

Get all unacknowledged alert messages for a user.

Parameters:

usernum the numerical id of the user for whom the alert messages will be fetched.

Returns:

NULL on failure, or a pointer to an `inx` holder containing all of the user's alert messages on success.

Definition at line 649 of file `datatier.c`.

References `alert_alloc()`, `inx_alloc()`, `inx_free()`, `inx_insert()`, `inx_t`, `log_error`, `log_info`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_alert_t`, `mm_free()`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_endpoint_alert_list()`.

5.154.1.9 `inx_t* meta_data_fetch_all_tags (uint64_t usernum)`

Fetch all the tags attached to messages of a specified user from the database.

Parameters:

usernum the numerical id of the user for whom the message tags will be fetched.

Returns:

NULL on failure, or an `inx` holder containing a list of all the user's messages' tags as managed strings on success.

Definition at line 1170 of file `datatier.c`.

References `inx_alloc()`, `inx_free()`, `inx_insert()`, `inx_t`, `log_error`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `res_field_string()`, `res_row_next()`, `res_table_free()`, `st_free()`, `stmt_get_result()`, `multi_t::type`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_endpoint_messages_tags()`.

5.154.1.10 `bool_t meta_data_fetch_folders (meta_user_t * user)`

Retrieve all of a user's message folders from the database.

Note:

If the user already had a working set of message folders they will be deleted first.

Parameters:

user a pointer to the meta user object of the user making the request, which will be updated on success.

Returns:

-1 on failure or 1 on success.

Definition at line 58 of file `datatier.c`.

References `inx_alloc()`, `inx_cleanup()`, `inx_insert()`, `log_error`, `log_pedantic`, `M_FOLDER_MESSAGES`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_folder_t`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `mm_wipe()`, `res_field_block()`, `res_field_length()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::type`, `type()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `meta_update_folders()`.

5.154.1.11 `int_t meta_data_fetch_keys (meta_user_t * user, key_pair_t * output, int64_t transaction)`

Retrieve the encryption keys for a user account.

Parameters:

user a meta object with the user number field populated.

output a key pair object to hold the results.

transaction the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

-1 for unexpected program/system error, 0 on success, 1 if no rows are found.

Definition at line 307 of file `datatier.c`.

References `base64_decode_mod()`, `log_pedantic`, `mm_wipe()`, `PLACER`, `key_pair_t::private`, `key_pair_t::public`, `res_field_block()`, `res_field_length()`, `res_row_next()`, `res_table_free()`, `st_char_get()`, `st_empty`, `st_free()`, `st_length_int()`, and `stmt_get_result_conn()`.

Referenced by `meta_update_keys()`.

5.154.1.12 int_t meta_data_fetch_mailbox_aliases (meta_user_t * user)

Get all the mailbox aliases for a specified user.

Parameters:

user a pointer to the partially populated meta user object to be queried, with its usernum field set.

Returns:

true on success or false on failure.

Definition at line 159 of file datatier.c.

References alias_alloc(), inx_alloc(), inx_cleanup(), inx_insert(), log_error, log_info, log_pedantic, M_INX_LINKED, M_TYPE_UINT64, meta_alias_t, mm_free(), mm_wipe(), PLACER, res_field_block(), res_field_length(), res_field_uint64(), res_field_uint8(), res_row_next(), res_table_free(), st_char_get(), st_length_int(), stmt_get_result(), multi_t::u64, and multi_t::val.

Referenced by meta_update_aliases().

5.154.1.13 int_t meta_data_fetch_shard (uint64_t usernum, uint16_t serial, stringer_t * label, stringer_t * output, uint_t * rotated, int64_t transaction)

Retrieve a shard value for user account.

Parameters:

usernum the numerical id of the user to whom the alert message belongs.

serial the serial number associated with the shard value.

label the textual label associated with the shard value.

output the buffer where the binary output should be stored.

rotated the buffer where the rotated variable is stored.

transaction the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

-1 for unexpected program/system error, 0 on success, 1 if no rows are found.

Definition at line 440 of file datatier.c.

References base64_decode_mod(), log_pedantic, mm_wipe(), PLACER, res_field_block(), res_field_length(), res_field_uint8(), res_row_next(), res_table_free(), st_avail_get(), st_char_get(), st_data_get(), st_empty, st_length_get(), st_length_int(), st_opt_get(), st_valid_destination(), STACIE_SHARD_LENGTH, and stmt_get_result_conn().

Referenced by meta_update_realms().

5.154.1.14 int_t meta_data_fetch_user (meta_user_t * user)

Build a meta user object by username, hashed password, and hashed key storage password.

Parameters:

user a meta object with the user number field populated.

Returns:

-1 for unexpected program/system error, 0 on success, 1 if no rows are found.

Definition at line 224 of file `datatier.c`.

References `base64_decode_mod()`, `log_pedantic`, `META_USER_ENCRYPT_DATA`, `META_USER_OVERQUOTA`, `META_USER_TLS`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_int8()`, `res_field_length()`, `res_field_string()`, `res_row_next()`, `res_table_free()`, `st_cleanup`, `st_empty`, and `stmt_get_result()`.

Referenced by `meta_update_user()`.

5.154.1.15 `bool_t meta_data_flags_add (inx_t * messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)`

Add the specified flags mask to a collection of mail messages.

Parameters:

messages an `inx` holder containing the collection of messages to have their flags updated.

usernum the numerical id of the user to whom the target messages belong, for validation purposes.

foldernum the numerical id of the parent folder containing the messages to be updated, for validation purposes.

flags a mask of all flags that are to be added to any matching messages in the collection.

Returns:

true on success or false on failure.

Definition at line 869 of file `datatier.c`.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `meta_message_t`, `mm_wipe()`, and `stmt_exec()`.

Referenced by `imap_fetch()`, `imap_update_flags()`, `portal_endpoint_messages_flag()`, and `portal_endpoint_messages_tag()`.

5.154.1.16 `bool_t meta_data_flags_remove (inx_t * messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)`

Remove the specified flags mask from a collection of mail messages.

Parameters:

messages an `inx` holder containing the collection of messages to have their flags removed.

usernum the numerical id of the user to whom the target messages belong, for validation purposes.

foldernum the numerical id of the parent folder containing the messages to be updated, for validation purposes.

flags a mask of all flags that are to be stripped from any matching messages in the collection.

Returns:

true on success or false on failure.

Definition at line 795 of file `datatier.c`.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `meta_message_t`, `mm_wipe()`, and `stmt_exec()`.

Referenced by `imap_select()`, `imap_update_flags()`, `portal_endpoint_messages_flag()`, and `portal_endpoint_messages_tag()`.

5.154.1.17 `bool_t meta_data_flags_replace (inx_t * messages, uint64_t usernum, uint64_t foldernum, uint32_t flags)`

Remove all user (non-system) flags from a collection of mail messages, and set the specified flags mask for them.

Note:

The new mask can contain both user and system flags, but only user flags will be stripped from each message initially.

Parameters:

messages an inx holder containing the collection of messages to have their flags updated.
usenum the numerical of the user to whom the target messages belong, for validation purposes.
foldernum the numerical id of the parent folder containing the messages to be updated, for validation purposes.
flags a mask of all flags that are to be added to any matching messages in the collection.

Returns:

true on success or false on failure.

Definition at line 713 of file datatier.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `MAIL_STATUS_USER_FLAGS`, `meta_message_t`, `mm_wipe()`, and `stmt_exec()`.

Referenced by `imap_update_flags()`, and `portal_endpoint_messages_flag()`.

5.154.1.18 uint64_t meta_data_insert_folder (uint64_t usenum, stringer_t * name, uint64_t parent, uint32_t order)

Insert a new mail folder into the database.

Parameters:

usenum the numerical id of the user to whom the new folder belongs.
name a managed string containing the name of the new mail folder.
parent the numerical id of the mail folder to be the parent of the new mail folder.
order the order number of this folder in its parent folder.

Returns:

0 on failure, or the numerical id of the newly inserted mail folder in the database on success.

Definition at line 1028 of file datatier.c.

References `M_FOLDER_MESSAGES`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `stmt_insert()`, and `type()`.

Referenced by `imap_folder_create()`, and `imap_folder_rename()`.

5.154.1.19 int_t meta_data_insert_keys (uint64_t usenum, stringer_t * username, key_pair_t * input, int64_t transaction)

Store the encryption keys for a user account.

Parameters:

usenum the numerical id of the user to whom the alert message belongs.
username the plain text username.
input the public and private key pair.
transaction the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

-1 for unexpected program/system error, 0 on success, 1 if the query executes, but no rows are affected.

Definition at line 372 of file datatier.c.

References `base64_encode_mod()`, `log_pedantic`, `mm_wipe()`, `key_pair_t::private`, `key_pair_t::public`, `st_char_get()`, `st_cleanup`, `st_length_get()`, `st_length_int()`, `st_populated`, and `stmt_exec_affected_conn()`.

Referenced by `meta_crypto_keys_create()`.

5.154.1.20 `int_t meta_data_insert_shard (uint64_t usernum, uint16_t serial, stringer_t * label, stringer_t * shard, int64_t transaction)`

Store the user realm shard value.

Parameters:

usernum the numerical id of the user to whom the alert message belongs.

serial the nummerid serial number associated with the shard value.

label the textual label associated with the shard value.

shard the binary shard value.

transaction the mysql transaction id of the acknowledgment operation, in cases batch changes need to be rolled back.

Returns:

-1 for unexpected program/system error, 0 on success, 1 if the query executes, but no rows are affected.

Definition at line 508 of file `datatier.c`.

References `base64_encode_mod()`, `log_pedantic`, `MANAGEDBUF`, `mm_wipe()`, `st_char_get()`, `st_data_get()`, `st_empty`, `st_length_get()`, `st_length_int()`, `STACIE_SHARD_LENGTH`, and `stmt_exec_affected_conn()`.

Referenced by `meta_update_realms()`.

5.154.1.21 `int_t meta_data_insert_tag (meta_message_t * message, stringer_t * tag)`

Insert a tag for a message into the database.

Parameters:

message the meta message object of the message to be tagged.

tag a managed string containing the name of the tag.

Returns:

0 on success or -1 on failure.

Definition at line 1075 of file `datatier.c`.

References `mm_wipe()`, `st_char_get()`, `st_length_get()`, and `stmt_exec_affected()`.

Referenced by `portal_endpoint_messages_tag()`.

5.154.1.22 `int_t meta_data_truncate_tags (meta_message_t * message)`

Remove all tags associated with a message in the database.

Parameters:

message a pointer to the meta message object of the message to have all of its tags stripped.

Returns:

0 on success or -1 on failure.

Definition at line 1108 of file `datatier.c`.

References `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `portal_endpoint_messages_tag()`.

5.154.1.23 `uint64_t meta_data_update_folder_name (uint64_t usernum, uint64_t foldernum, stringer_t * name, uint64_t parent, uint32_t order)`

Update the record for a message folder in the database.

Parameters:

- usernum* the numerical id of the user that owns the specified folder.
- foldernum* the id of the folder to have its properties adjusted.
- name* a managed string containing the new name of the specified folder.
- parent* the id of the new parent folder to be set for the specified message folder.
- order* the value of the order for the specified folder.

Returns:

1 on success, or <= 0 on failure.

Definition at line 972 of file `datatier.c`.

References `M_FOLDER_MESSAGES`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `stmt_exec_affected()`, and `type()`.

Referenced by `imap_folder_rename()`.

5.154.1.24 `void meta_data_update_lock (uint64_t usernum, uint8_t lock)`

5.154.1.25 `void meta_data_update_log (meta_user_t * user, META_PROTOCOL prot)`

Update the per-user entry in the Log table for the specified protocol.

Note:

This function will set the last session timestamp for the user, and increment the sessions counter in the database.

Parameters:

- user* the meta user object of the user making the logging request.
- prot* the protocol associated with the log request: `META_PROT_POP`, `META_PROT_IMAP`, or `META_PROT_WEB`.

Returns:

This function returns no value.

Definition at line 20 of file `datatier.c`.

References `log_pedantic`, `META_PROTOCOL_IMAP`, `META_PROTOCOL_POP`, `META_PROTOCOL_WEB`, `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `imap_login()`, and `pop_pass()`.

5.154.1.26 `meta_stats_tag_t* meta_folder_stats_tag_alloc (stringer_t * tag)`

`folders.c` `folders.c`

Parameters:

- tag* a managed string containing the name of the message tag.

Returns:

NULL on failure, or a newly allocated and initialized meta tag stat object on success.

Definition at line 15 of file folders.c.

References align(), FOREIGNDATA, JOINTED, log_pedantic, mm_alloc(), mm_copy(), PLACER_T, placer_t, st_data_get(), st_length_get(), STACK, and meta_stats_tag_t::tag.

Referenced by meta_folders_stats_tags().

5.154.1.27 meta_folder_t* meta_folders_by_name (inx_t * *folders*, stringer_t * *name*)

Get a folder by its fully qualified name.

Parameters:

folders a pointer to an inx holder containing a list of folders to be traversed in the search.

folder a pointer to the meta folder object that is the leaf node of the folder path.

Returns:

NULL on failure or a managed string containing the fully qualified folder name on success.

Definition at line 138 of file folders.c.

References CONSTANT, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), meta_folder_t, meta_folders_name(), st_cmp_ci_eq(), st_cmp_cs_eq(), and st_free().

Referenced by imap_append(), imap_copy(), imap_folder_create(), imap_folder_remove(), imap_folder_rename(), imap_folder_status(), imap_subscribe(), and portal_folder_mail_add().

5.154.1.28 meta_folder_t* meta_folders_by_number (inx_t * *folders*, uint64_t *number*)

Get a folder by number.

Parameters:

folders a pointer to an inx holder containing the folders to be searched.

number the numerical id of the folder to be retrieved.

Returns:

NULL on failure, or a pointer to the found meta folder object of the specified folder on success.

Definition at line 179 of file folders.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), and meta_folder_t.

Referenced by meta_folders_name(), portal_endpoint_folders_rename(), portal_endpoint_folders_tags(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_tag(), portal_folder_mail_add(), and portal_folder_mail_remove().

5.154.1.29 int_t meta_folders_children (inx_t * *folders*, uint64_t *number*)

Get the number of direct child folders of a specified parent folder.

Parameters:

folders a pointer to an inx holder containing the the collection of folders to be traversed.

number the folder id of the parent folder to be scanned for children.

Returns:

the number of children folders in the specified parent folder, or 0 on failure.

Definition at line 207 of file folders.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, and `meta_folder_t`.

Referenced by `imap_folder_remove()`.

5.154.1.30 `stringer_t* meta_folders_name (inx_t * list, meta_folder_t * folder)`

Get a folder's fully qualified name.

Note:

This function will prepend the entire ancestor path of a folder to its name, with each level delimited by periods.

Parameters:

list a pointer to an `inx` holder containing a list of folders to be searched for parent folders.

folder a pointer to the meta folder object that is the leaf node of the folder path.

Returns:

NULL on failure or a managed string containing the fully qualified folder name on success.

Definition at line 99 of file folders.c.

References `FOLDER_RECURSION_LIMIT`, `log_pedantic`, `meta_folders_by_number()`, `ns_length_get()`, `st_free()`, `st_import()`, and `st_merge`.

Referenced by `imap_folder_name_escaped()`, `imap_narrow_folders()`, `meta_folders_by_name()`, `portal_endpoint_folders_rename()`, `portal_folder_mail_add()`, and `portal_folder_mail_remove()`.

5.154.1.31 `inx_t* meta_folders_stats_tags (inx_t * messages, uint64_t folder)`

Create a collection of meta tag stats for a set of messages.

Note:

Each meta tag stat will contain the name of a message tag, along with the number of messages with which it was associated.

Parameters:

messages an `inx` holder containing the list of messages to be examined.

folder the numerical id of the parent folder that contains all target messages.

Returns:

NULL on failure, or an `inx` holder containing all of the messages' meta tag stats on success.

LOW: This is a very inefficient method for counting the number of times each tag appears.

Definition at line 40 of file folders.c.

References `ar_field_st()`, `ar_length_get()`, `meta_stats_tag_t::count`, `inx_alloc()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_find()`, `inx_insert()`, `inx_t`, `log_pedantic`, `M_INX_HASHED`, `M_TYPE_STRINGER`, `meta_folder_stats_tag_alloc()`, `meta_message_t`, `mm_free()`, `multi_t::st`, and `multi_t::val`.

Referenced by `portal_endpoint_folders_tags()`.

5.154.1.32 void meta_free (meta_user_t * user)

Free a meta user object.

Parameters:

user a pointer to the meta user object to be destroyed.

Returns:

This function returns no value.

Definition at line 17 of file meta.c.

References `inx_cleanup()`, `mm_free()`, `mutex_destroy()`, `prime_cleanup()`, `rwlock_destroy()`, and `st_cleanup`.

Referenced by `meta_inx_find()`, and `obj_cache_start()`.

5.154.1.33 int_t meta_get (uint64_t usernum, stringer_t * username, stringer_t * salt, stringer_t * master, stringer_t * verification, META_PROTOCOL protocol, META_GET get, meta_user_t ** output)

Lookup user and return their meta user object.

Note:

If the user is not found in the local session cache, the session will be constructed using the database, and then cached.

Parameters:

usernum the numeric identifier for the user account.

username the official username stored in the database.

salt the user specific salt value.

master the user account's master encryption key which will be used to unlock the private storage key.

verification the verification token.

protocol a set of protocol specifying the protocol used by the calling function. Values can be `META_PROT_NONE`, `META_PROT_SMTP`, `META_PROT_POP`, `META_PROT_IMAP`, `META_PROT_WEB`, or `META_PROT_GENERIC`.

get a set of protocol specifying the data to be retrieved (`META_GET_NONE`, `META_GET_MESSAGES`, `META_GET_FOLDERS`, or `META_GET_CONTACTS`)

output the address of a meta user object that will store a pointer to the result of the lookup.

Returns:

-1 on error, 0 on success, 1 for an authentication issue.

Definition at line 112 of file meta.c.

References `log_pedantic`, `META_GET_ALIASES`, `META_GET_CONTACTS`, `META_GET_FOLDERS`, `META_GET_KEYS`, `META_GET_MESSAGES`, `meta_inx_find()`, `meta_inx_remove()`, `META_LOCKED`, `meta_messages_update()`, `meta_update_aliases()`, `meta_update_contacts()`, `meta_update_folders()`, `meta_update_keys()`, `meta_update_message_folders()`, `meta_update_realms()`, `meta_update_user()`, `meta_user_t`, `meta_user_unlock()`, `meta_user_wlock()`, `st_cleanup`, `st_cmp_cs_eq()`, `st_empty`, and `st_populated`.

Referenced by `api_endpoint_auth()`, `imap_login()`, `pop_pass()`, and `portal_endpoint_auth()`.

5.154.1.34 meta_user_t* meta_inx_find (uint64_t usernum, META_PROTOCOL protocol)

[indexes.c](#)

Definition at line 40 of file indexes.c.

References `inx_find()`, `inx_insert()`, `inx_lock_write()`, `inx_unlock()`, `M_TYPE_UINT64`, `object_cache_t::meta`, `meta_alloc()`, `meta_free()`, `meta_user_ref_add()`, `meta_user_t`, and `objects`.

Referenced by `meta_get()`.

5.154.1.35 `void meta_inx_remove(uint64_t usernum, META_PROTOCOL protocol)`

Lock a user's object in the cache and decrement their reference counter.

See also:

[meta_user_ref_dec\(\)](#)

Parameters:

username a managed string containing the name of the user to be adjusted.

protocol specifies the protocol bound to the reference counter to be decremented (`META_PROT_WEB`, `META_PROT_IMAP`, etc.)

Returns:

This function returns no value.

Definition at line 17 of file `indexes.c`.

References `inx_find()`, `inx_lock_read()`, `inx_unlock()`, `M_TYPE_UINT64`, `object_cache_t::meta`, `meta_user_ref_dec()`, `meta_user_t`, and `objects`.

Referenced by `imap_login()`, `imap_session_destroy()`, `meta_get()`, `pop_pass()`, `pop_session_destroy()`, `portal_endpoint_auth()`, and `sess_destroy()`.

5.154.1.36 `int_t meta_update_aliases(meta_user_t * user, META_LOCK_STATUS locked)`

[updaters.c](#) [updaters.c](#)

Parameters:

user a pointer to the meta object that is to be populated.

locked the meta lock status of the operation (if `META_NEED_LOCK` is supplied, the meta user object will be locked for the duration of the function).

Returns:

Definition at line 261 of file `updaters.c`.

References `meta_data_fetch_mailbox_aliases()`, `META_NEED_LOCK`, `meta_user_serial_get()`, `meta_user_serial_set()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_ALIASES`, `serial_get()`, and `serial_increment()`.

Referenced by `meta_get()`, and `sess_update()`.

5.154.1.37 `int_t meta_update_contacts(meta_user_t * user, META_LOCK_STATUS locked)`

Update a user's contacts if necessary.

Note:

This function will try to retrieve the folders from the cache, if possible, or fall back to the database.

Parameters:

- user* a pointer to the meta user object that will have its message folders updated.
- locked* if META_NEED_LOCK is specified, the meta user object will be locked for operation.

Returns:

-1 on failure or 1 on success.

Definition at line 310 of file updaters.c.

References contacts_update(), inx_free(), inx_t, META_NEED_LOCK, meta_user_unlock(), meta_user_wlock(), OBJECT_CONTACTS, serial_get(), and serial_increment().

Referenced by meta_get(), and sess_update().

5.154.1.38 int_t meta_update_folders (meta_user_t * user, META_LOCK_STATUS locked)

Update a user's message folders if necessary.

Note:

This function will try to retrieve the folders from the cache, if possible, or fall back to the database.

Parameters:

- user* a pointer to the meta user object that will have its message folders updated.
- locked* if META_NEED_LOCK is specified, the meta user object will be locked for operation.

Returns:

-1 on failure or 1 on success.

Definition at line 367 of file updaters.c.

References meta_data_fetch_folders(), meta_messages_update_sequences(), META_NEED_LOCK, meta_user_unlock(), meta_user_wlock(), OBJECT_FOLDERS, serial_get(), and serial_increment().

Referenced by imap_session_update(), meta_get(), and sess_update().

5.154.1.39 int_t meta_update_keys (meta_user_t * user, META_LOCK_STATUS locked)

Fetches the user signet and private key. Relies on the mail realm to decrypt the values.

Parameters:

- user* a pointer to the meta object that is to be populated.
- locked* the meta lock status of the operation (if META_NEED_LOCK is supplied, the meta user object will be locked for the duration of the function).

Returns:

-2 if there is a problem unscrambling the private key, -1 for a system error, 0 for success, and 1 if the keys were created.

Definition at line 113 of file updaters.c.

References BINARY, log_pedantic, meta_crypto_keys_create(), meta_data_fetch_keys(), META_NEED_LOCK, meta_user_unlock(), meta_user_wlock(), NONE, prime_key_decrypt(), prime_set(), key_pair_t::private, key_pair_t::public, st_char_get(), st_cleanup, st_length_int(), st_populated, tran_commit(), tran_rollback(), and tran_start().

Referenced by meta_get().

5.154.1.40 int_t meta_update_message_folders (meta_user_t * user, META_LOCK_STATUS locked)

Refresh a user's message folders if stale, or retrieve them from the database if they are not in memory.

See also:

[messages_update\(\)](#)

Parameters:

user a pointer to the meta user object requesting the folders.

locked if META_LOCKED, lock the meta user object for the duration of the request.

Returns:

-1 on failure, or 1 on success.

Definition at line 419 of file updaters.c.

References [inx_free\(\)](#), [inx_t](#), [messages_update\(\)](#), [META_NEED_LOCK](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_FOLDERS](#), [serial_get\(\)](#), and [serial_increment\(\)](#).

Referenced by [meta_get\(\)](#).

5.154.1.41 int_t meta_update_realms (meta_user_t * user, stringer_t * salt, stringer_t * master, META_LOCK_STATUS locked)

Fetches the user realm keys and extracts the different components.

Parameters:

user a pointer to the meta object that is to be populated.

master the user master key value.

locked the meta lock status of the operation (if META_NEED_LOCK is supplied, the meta user object will be locked for the duration of the function).

Returns:

-2 if there is a problem extracting the key values, -1 for a system error, 0 for success, and 1 if the keys were created.

****/

****/

Definition at line 20 of file updaters.c.

References [log_pedantic](#), [MANAGEDBUF](#), [meta_data_fetch_shard\(\)](#), [meta_data_insert_shard\(\)](#), [META_NEED_LOCK](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [PLACER](#), [st_char_get\(\)](#), [st_empty](#), [st_length_get\(\)](#), [st_length_int\(\)](#), [stacie_create_shard\(\)](#), [STACIE_KEY_LENGTH](#), [stacie_realm_key\(\)](#), [STACIE_SHARD_LENGTH](#), [tran_commit\(\)](#), [tran_rollback\(\)](#), and [tran_start\(\)](#).

Referenced by [meta_get\(\)](#).

5.154.1.42 int_t meta_update_user (meta_user_t * user, META_LOCK_STATUS locked)

Build a user's meta information from specified data parameters.

Note:

The user object will be pulled from the cache, if possible, or it falls back to a database lookup using the user number and the verification token.

Parameters:

user a pointer to the meta object that is to be populated.

usernum the user number.

verification the verification token for the specified user number.

locked the meta lock status of the operation (if META_NEED_LOCK is supplied, the meta user object will be locked for the duration of the function).

Returns:

-1 on error, 0 on success, 1 for an authentication issue.

Definition at line 211 of file updaters.c.

References `meta_data_fetch_user()`, `META_NEED_LOCK`, `meta_user_serial_get()`, `meta_user_serial_set()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_USER`, `serial_get()`, `serial_increment()`, and `st_populated`.

Referenced by `imap_session_update()`, `meta_get()`, and `sess_update()`.

5.154.1.43 void meta_user_ref_add (meta_user_t * user, META_PROTOCOL protocol)

[references.c](#) [references.c](#)

Note:

META_PROTOCOL_GENERIC can be specified for non-specific, protocol independent accounting purposes.

Parameters:

user a pointer to the meta user object to be adjusted.

protocol the protocol identifier for the session.

Returns:

This function returns no value.

Definition at line 20 of file references.c.

References `log_pedantic`, `META_PROTOCOL_GENERIC`, `META_PROTOCOL_IMAP`, `META_PROTOCOL_POP`, `META_PROTOCOL_SMTP`, `META_PROTOCOL_WEB`, `mutex_lock()`, and `mutex_unlock()`.

Referenced by `meta_inx_find()`.

5.154.1.44 void meta_user_ref_dec (meta_user_t * user, META_PROTOCOL protocol)

Decrement a user's reference counter for the specified protocol and update the activity timestamp.

Note:

META_PROTOCOL_GENERIC can be specified for non-specific, protocol independent accounting purposes.

Parameters:

user a pointer to the meta user object to be adjusted.

protocol the protocol identifier for the session using the META_PROTOCOL enumerator.

Returns:

This function returns no value.

Definition at line 60 of file references.c.

References log_pedantic, META_PROTOCOL_GENERIC, META_PROTOCOL_IMAP, META_PROTOCOL_POP, META_PROTOCOL_SMTP, META_PROTOCOL_WEB, mutex_lock(), and mutex_unlock().

Referenced by meta_inx_remove(), and portal_endpoint_logout().

5.154.1.45 uint64_t meta_user_ref_protocol_total (meta_user_t * user, META_PROTOCOL protocol)

Get the total reference count for the user object over the specified protocol.

Parameters:

user a pointer to the meta user object to be examined.

protocol the protocol identifier for the session using the META_PROTOCOL enumerator.

Returns:

the total number of references held by the user.

Definition at line 98 of file references.c.

References log_pedantic, META_PROTOCOL_GENERIC, META_PROTOCOL_IMAP, META_PROTOCOL_POP, META_PROTOCOL_SMTP, META_PROTOCOL_WEB, mutex_lock(), and mutex_unlock().

Referenced by pop_pass().

5.154.1.46 time_t meta_user_ref_stamp (meta_user_t * user)

Get the activity timestamp for a meta user object.

Parameters:

user a pointer to the meta user object to be examined.

Returns:

a timestamp containing the last time the meta user object's reference count changed.

Definition at line 161 of file references.c.

References mutex_lock(), and mutex_unlock().

Referenced by obj_cache_prune().

5.154.1.47 uint64_t meta_user_ref_total (meta_user_t * user)

Get the total reference count for a user object, over all of the supported protocols.

Parameters:

user a pointer to the meta user object to be examined.

Returns:

the total number of references held by the user.

Definition at line 134 of file references.c.

References mutex_lock(), and mutex_unlock().

Referenced by obj_cache_prune().

5.154.1.48 void meta_user_rlock (meta_user_t * user)[locking.c locking.c](#)**Parameters:**

user a pointer to the meta user object to be locked.

Returns:

This function returns no value.

Definition at line 15 of file locking.c.

References `rwlock_lock_read()`.

Referenced by `imap_close()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_list()`, `imap_login()`, `imap_lsub()`, `imap_search_messages()`, `imap_status()`, `pop_last()`, `pop_list()`, `pop_retr()`, `pop_stat()`, `pop_top()`, `pop_uidl()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, and `portal_endpoint_messages_tag()`.

5.154.1.49 bool_t meta_user_serial_check (meta_user_t * user, uint64_t object)

[serials.c serials.c](#)

Note:

The object's serial number will be incremented regardless of whether it is consistent with the cache.

Parameters:

user the meta user object to whom the object belongs.

object the serial object to be checked for changes.

Returns:

0 if the object did not to be refreshed, or 1 if it doesn't match the internal checkpoint and should be updated.

Definition at line 86 of file serials.c.

References `meta_user_serial_get()`, `meta_user_serial_set()`, `serial_get()`, and `serial_increment()`.

Referenced by `portal_endpoint_messages_tag()`, and `sess_serial_check()`.

5.154.1.50 uint64_t meta_user_serial_get (meta_user_t * user, uint64_t object)

Get an object serial number for a given meta object.

Parameters:

user the meta user structure to be examined.

object the object to query for the serial number.

Returns:

the serial number value of the specified object, or 0 on failure. LOW: Swap the object and user params so the interface matches the `serial_get/set` functions.

Definition at line 53 of file serials.c.

References `OBJECT_ALIASES`, `OBJECT_CONTACTS`, `OBJECT_FOLDERS`, `OBJECT_MESSAGES`, and `OBJECT_USER`.

Referenced by `meta_update_aliases()`, `meta_update_user()`, and `meta_user_serial_check()`.

5.154.1.51 void meta_user_serial_set (meta_user_t * user, uint64_t object, uint64_t serial)

Set an object serial number for a given meta user structure.

Parameters:

user the meta user object to be adjusted.

object the object to receive the new serial number.

serial the new serial number to be set.

Returns:

This function returns no value.

Definition at line 20 of file serials.c.

References OBJECT_ALIASES, OBJECT_CONTACTS, OBJECT_FOLDERS, OBJECT_MESSAGES, and OBJECT_USER.

Referenced by meta_update_aliases(), meta_update_user(), and meta_user_serial_check().

5.154.1.52 void meta_user_unlock (meta_user_t * user)

Release the lock for a meta user object.

Parameters:

user a pointer to the meta user object to be unlocked.

Returns:

This function returns no value.

Definition at line 47 of file locking.c.

References rwlock_unlock().

Referenced by imap_append(), imap_close(), imap_copy(), imap_create(), imap_delete(), imap_examine(), imap_expunge(), imap_fetch(), imap_list(), imap_login(), imap_lsub(), imap_rename(), imap_search_messages(), imap_select(), imap_session_destroy(), imap_session_update(), imap_status(), imap_store(), imap_subscribe(), meta_get(), meta_messages_copier(), meta_messages_login_update(), meta_messages_mover(), meta_messages_update(), meta_update_aliases(), meta_update_contacts(), meta_update_folders(), meta_update_keys(), meta_update_message_folders(), meta_update_realms(), meta_update_user(), pop_dele(), pop_last(), pop_list(), pop_pass(), pop_retr(), pop_session_destroy(), pop_session_reset(), pop_stat(), pop_top(), pop_uidl(), portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_edit(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), portal_endpoint_folders_rename(), portal_endpoint_folders_tags(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_tag(), portal_folder_contacts_add(), portal_folder_contacts_remove(), portal_folder_mail_add(), and portal_folder_mail_remove().

5.154.1.53 void meta_user_wlock (meta_user_t * user)

Acquire a write lock for a meta user object.

Parameters:

user a pointer to the meta user object to be locked.

Returns:

This function returns no value.

Definition at line 31 of file locking.c.

References `rwlock_lock_write()`.

Referenced by `imap_append()`, `imap_close()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_rename()`, `imap_select()`, `imap_session_destroy()`, `imap_session_update()`, `imap_store()`, `imap_subscribe()`, `meta_get()`, `meta_messages_copier()`, `meta_messages_login_update()`, `meta_messages_mover()`, `meta_messages_update()`, `meta_update_aliases()`, `meta_update_contacts()`, `meta_update_folders()`, `meta_update_keys()`, `meta_update_message_folders()`, `meta_update_realms()`, `meta_update_user()`, `pop_dele()`, `pop_pass()`, `pop_session_destroy()`, `pop_session_reset()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_folders_rename()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_tag()`, `portal_folder_contacts_add()`, `portal_folder_contacts_remove()`, `portal_folder_mail_add()`, and `portal_folder_mail_remove()`.

5.155 src/network/network.h File Reference

```
#include "meta.h"
#include "sessions.h"
#include "pop.h"
#include "smtp.h"
#include "dmtip.h"
#include "imap.h"
#include "http.h"
```

Enumerations

- enum { [REVERSE_ERROR](#) = -1, [REVERSE_EMPTY](#) = 0, [REVERSE_PENDING](#) = 1, [REVERSE_COMPLETE](#) = 2 }

Functions

- struct [__attribute__](#) ((packed))
- [ip_t * con_addr](#) ([connection_t](#) *con, [ip_t](#) *output)
addresses.c
- [octet_t con_addr_octet](#) ([connection_t](#) *con, [int_t](#) position)
Get a specified 8-bit octet of a connection's peer IP address.
- [stringer_t * con_addr_presentation](#) ([connection_t](#) *con, [stringer_t](#) *output)
Return a textual representation of the IP address of a specified connection handle.
- [stringer_t * con_addr_reversed](#) ([connection_t](#) *con, [stringer_t](#) *output)
Get the reversed-IP address string for a client connection.
- [segment_t con_addr_segment](#) ([connection_t](#) *con, [int_t](#) position)
Extract a specified 16 bit segment from a connection's peer IP address.
- [stringer_t * con_addr_standard](#) ([connection_t](#) *con, [stringer_t](#) *output)
Get the IP address string for the client connection.
- [stringer_t * con_addr_subnet](#) ([connection_t](#) *con, [stringer_t](#) *output)
Get the subnet string for a specified connection.
- [uint32_t con_addr_word](#) ([connection_t](#) *con, [int_t](#) position)
Get a specified 32-bit segment of a connection's peer IP address.
- [uint64_t con_decrement_refs](#) ([connection_t](#) *con)
connections.c
- void [con_destroy](#) ([connection_t](#) *con)
Destroy and free a generic connection object after executing its protocol-specific destructor; update any statistics accordingly.
- void [con_flush](#) ([connection_t](#) *con)
Attempt to flush any buffered data associated with a network connection.

- `uint64_t con_increment_refs (connection_t *con)`
Increment a connection object's reference counter.
- `connection_t * con_init (int cond, server_t *server)`
Create a new connection object for a client connection.
- `bool_t con_init_network_buffer (connection_t *con)`
Allocate and initialize a new network buffer for a connection, if it is not already associated with one.
- `bool_t con_localhost (connection_t *con)`
Determine whether a client connection is from the same machine, using the loopback adapter, or a remote machine.
- `bool_t con_private (connection_t *con)`
Determine whether a client connection is with a machine on the same private network.
- `int_t con_secure (connection_t *con)`
Return the security level of a specified connection.
- `int_t con_status (connection_t *con)`
Return the status of a specified connection.
- `void client_close (client_t *client)`
clients.c
- `client_t * client_connect (chr_t *host, uint32_t port)`
Establish a network client connection to a remote host.
- `int_t client_secure (client_t *client)`
Establish a TLS connection with a network client instance.
- `int_t client_status (client_t *client)`
Get the status of a network client.
- `bool_t net_set_buffer_length (int sd, int buffer_recv, int buffer_send)`
options.c
- `bool_t net_set_keepalive (int sd, bool_t keepalive, int_t idle, int_t interval, int_t tolerance)`
Set the keepalive flag, the.
- `bool_t net_set_linger (int sd, bool_t linger, int_t timeout)`
Set the linger flag for a socket.
- `bool_t net_set_nodelay (int sd, bool_t nodelay)`
Set the delay flag for a socket (to disable the Nagle algorithm).
- `bool_t net_set_blocking (int sd, bool_t blocking)`
Set the blocking flag for a socket.
- `bool_t net_set_reuseable_address (int sd, bool_t reuse)`
Set the reusable flag for a socket.
- `bool_t net_set_timeout (int sd, uint32_t timeout_recv, uint32_t timeout_send)`
Set the send and receive timeouts for a socket.

- `int64_t client_read (client_t *client)`
read.c
- `int64_t client_read_line (client_t *client)`
Read a line of input from a network client session.
- `int64_t con_read (connection_t *con)`
Read data from a network connection, and store the data in the connection context buffer.
- `int64_t con_read_line (connection_t *con, bool_t block)`
Read a line of input from a network connection.
- `stringer_t * con_reverse_check (connection_t *con, uint32_t timeout)`
reverse.c
- `void con_reverse_domain (connection_t *con, stringer_t *domain, int_t status)`
Set the domain name and reverse lookup status of a connection.
- `void con_reverse_enqueue (connection_t *con)`
Queue a reverse DNS lookup on the specified connection, if one hasn't been performed.
- `void con_reverse_lookup (connection_t *con)`
Perform a reverse DNS lookup on the remote end of a connection, and save the hostname.
- `void con_reverse_status (connection_t *con, int_t status)`
- `bool_t net_init (server_t *server)`
listeners.c
- `void net_listen (void)`
- `void net_shutdown (server_t *server)`
Close the listening socket associated with a server.
- `void net_trigger (bool_t verbose)`
Triggers a connection for each server instance, allowing the the listeners to shutdown cleanly. And then purges any remaining sockets.
- `int64_t client_print (client_t *client, chr_t *format,...)`
write.c
- `int64_t client_write (client_t *client, stringer_t *s)`
Write data to a network client.
- `int64_t con_print (connection_t *con, chr_t *format,...)`
Write a formatted string to a network connection.
- `int64_t con_write_bl (connection_t *con, char *block, size_t length)`
Write data to a network connection.
- `int64_t con_write_ns (connection_t *con, char *string)`
Write a null-terminated string to a network connection.
- `int64_t con_write_pl (connection_t *con, placer_t string)`
Write a placer to a network connection.

- `int64_t con_write_st (connection_t *con, stringer_t *string)`

Write a managed string to a network connection.

- `stringer_t * protocol_type (connection_t *con)`

Variables

- `command_t`
- `client_t`
- `connection_t`

5.155.1 Enumeration Type Documentation

5.155.1.1 anonymous enum

Enumerator:

REVERSE_ERROR

REVERSE_EMPTY

REVERSE_PENDING

REVERSE_COMPLETE

Definition at line 19 of file network.h.

5.155.2 Function Documentation

5.155.2.1 `struct __attribute__((packed)) [read]`

Definition at line 43 of file network.h.

References `command`, `command_t`, `imap_session_t`, `lock`, `placer_t`, and `status`.

5.155.2.2 `void client_close (client_t * client)`

[clients.c](#) [clients.c](#)

Note:

If `ssl` is in use, this will also destroy the overlying `ssl` session.

Returns:

This function returns no value.

Definition at line 175 of file `clients.c`.

References `mm_cleanup`, `mm_free()`, `st_cleanup`, and `tls_free()`.

Referenced by `net_trigger()`, `smtp_client_close()`, and `smtp_client_connect()`.

5.155.2.3 client_t* client_connect (chr_t * *host*, uint32_t *port*)

Establish a network client connection to a remote host.

Parameters:

host a pointer to a null-terminated string containing the hostname of the remote server.

port the port number of the server to which the connection will be established.

Returns:

NULL on failure or a pointer to a newly initialized network client object for the connection upon success.

Definition at line 70 of file clients.c.

References client_t, ip_t::family, FOREIGNDATA, ip_t::ip4, ip_t::ip6, JOINTED, log_pedantic, MEMORYBUF, mm_alloc(), mm_cleanup, mm_copy(), mm_free(), PLACER_T, st_alloc(), STACK, and status.

Referenced by net_trigger(), and smtp_client_connect().

5.155.2.4 int64_t client_print (client_t * *client*, chr_t * *format*, ...)

[write.c](#) [write.c](#)

See also:

[client_write\(\)](#)

Parameters:

client the network client connection to which the supplied data will be written.

format the format string for the data to be written to the network client connection.

... a va_arg style collection of variables to be expanded by the passed format string.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 258 of file write.c.

References client_write(), st_free(), and st_vaprint.

Referenced by smtp_client_send_helo(), smtp_client_send_mailfrom(), smtp_client_send_nullfrom(), and smtp_client_send_rcptto().

5.155.2.5 int64_t client_read (client_t * *client*)

[read.c](#) [read.c](#)

Returns:

-1 on general failure, -2 if the connection was reset, or the amount of data that was read.

Definition at line 304 of file read.c.

References bytes, client_status(), ip_presentation(), log_pedantic, MANAGEDBUF, MEMORYBUF, mm_move(), pl_length_get(), pl_null(), st_avail_get(), st_char_get(), st_data_get(), st_length_get(), st_length_int(), st_length_set(), status, tcp_status(), tls_cipher(), tls_error(), and tls_read().

5.155.2.6 int64_t client_read_line (client_t * client)

Read a line of input from a network client session.

Returns:

-1 on general failure, -2 if the connection was reset, or the length of the current line of input, including the trailing new line character.

Definition at line 191 of file read.c.

References bytes, ip_presentation(), line_pl_st(), log_pedantic, MANAGEDBUF, MEMORYBUF, mm_move(), pl_empty(), pl_length_get(), pl_null(), st_avail_get(), st_char_get(), st_data_get(), st_empty, st_length_get(), st_length_int(), st_length_set(), status, tcp_status(), tls_cipher(), tls_error(), and tls_read().

Referenced by smtp_client_close(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), smtp_client_send_nullfrom(), and smtp_client_send_rcptto().

5.155.2.7 int_t client_secure (client_t * client)

Establish a TLS connection with a network client instance.

Parameters:

client a pointer to the network client object to have its transport security upgraded.

Returns:

-1 on failure or 0 on success.

Definition at line 45 of file clients.c.

References tls_client_alloc().

Referenced by smtp_client_connect().

5.155.2.8 int_t client_status (client_t * client)

Get the status of a network client.

Parameters:

client a pointer to the network client object to be queried.

Returns:

-1 on network errors, 0 for an unknown status, 1 for connected, and 2 for a graceful shutdown.

Definition at line 15 of file clients.c.

References tcp_status(), and tls_status().

Referenced by client_read(), and client_write().

5.155.2.9 int64_t client_write (client_t * client, stringer_t * s)

Write data to a network client.

Note:

This function works regardless of whether or not the client connection is ssl-enabled. If the network write requires multiple system calls, then this code will loop until all the data has been transmitted.

Parameters:

- client* the network client connection to which the supplied data will be written.
- s* a managed string containing the data to be written to the network client connection.

Returns:

- 1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 169 of file write.c.

References bytes, client_status(), ip_presentation(), length, log_pedantic, MANAGEDBUF, MEMORYBUF, st_char_get(), st_empty_out(), st_length_int(), status, tcp_status(), tls_cipher(), tls_error(), and tls_write().

Referenced by client_print(), smtp_client_close(), and smtp_client_send_data().

5.155.2.10 ip_t* con_addr (connection_t * con, ip_t * output)

[addresses.c](#) [addresses.c](#)

Parameters:

- con* the input client connection.
- output* a pointer to an [ip_t](#) structure to receive the remote IP address, which is allocated for the caller if output is NULL.

Returns:

- NULL on error, or a pointer to the IP address of the remote host.

Definition at line 16 of file addresses.c.

References ip_t::ip.

Referenced by con_addr_octet(), con_addr_presentation(), con_addr_reversed(), con_addr_segment(), con_addr_standard(), con_addr_subnet(), con_addr_word(), sess_create(), sess_get(), smtp_bypass_check(), and smtp_rcpt_to().

5.155.2.11 octet_t con_addr_octet (connection_t * con, int_t position)

Get a specified 8-bit octet of a connection's peer IP address.

Parameters:

- con* a pointer to the connection object to be examined.
- position* a zero-based index into the 8-bit octets that comprise the target IP address.

Returns:

- 1 on error, or the specified 8-bit octet of the passed address on success.

Definition at line 152 of file addresses.c.

References con_addr(), and ip_octet().

5.155.2.12 stringer_t* con_addr_presentation (connection_t * con, stringer_t * output)

Return a textual representation of the IP address of a specified connection handle.

Parameters:

- con* the remote connection to be queried.

output a managed string to store the result, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to a managed string containing a textual representation of the IP address.

Definition at line 62 of file addresses.c.

References con_addr(), and ip_presentation().

Referenced by api_endpoint_auth(), contact_abuse_checks(), contact_abuse_increment_history(), contact_business(), imap_id(), imap_login(), mail_add_inbound_headers(), mail_add_outbound_headers(), pop_pass(), portal_endpoint_auth(), portal_meta(), protocol_secure(), register_abuse_checks(), register_data_insert_user(), register_session_cache(), register_session_get(), smtp_auth_login(), smtp_auth_plain(), and smtp_rcpt_to().

5.155.2.13 stringer_t* con_addr_reversed (connection_t * con, stringer_t * output)

Get the reversed-IP address string for a client connection.

Parameters:

con the client connection to be queried.

output a managed string to receive the output, which will be allocated for the caller if passed as NULL.

Returns:

NULL on failure, or a pointer to a managed string containing a textual representation of the IP address on success.

Definition at line 98 of file addresses.c.

References con_addr(), and ip_reversed().

Referenced by smtp_check_greylist(), and smtp_check_rbl().

5.155.2.14 segment_t con_addr_segment (connection_t * con, int_t position)

Extract a specified 16 bit segment from a connection's peer IP address.

Parameters:

con a pointer to the connection object to be examined.

position the zero-indexed (starting at least-significant word) 16-bit segment of the IP address to be evaluated.

Returns:

-1 on failure, or the 32 bit-widened segment extracted from the supplied IP address.

Definition at line 170 of file addresses.c.

References con_addr(), and ip_segment().

5.155.2.15 stringer_t* con_addr_standard (connection_t * con, stringer_t * output)

Get the IP address string for the client connection.

Parameters:

con the client connection to be queried.

output a managed string to receive the output, which will be allocated for the caller if output is NULL.

Returns:

NULL on failure, or a pointer to a managed string containing a textual representation of the IP address.

Definition at line 80 of file addresses.c.

References `con_addr()`, and `ip_standard()`.

Referenced by `register_abuse_check_blocklist()`, `register_abuse_checks()`, and `register_abuse_increment_history()`.

5.155.2.16 stringer_t* con_addr_subnet (connection_t * con, stringer_t * output)

Get the subnet string for a specified connection.

Parameters:

con the client connection to be queried.

output a managed string that will store the output of the subnet string lookup.

Returns:

NULL on failure, or a managed string containing a textual representation of a subnet address on success.

Definition at line 116 of file addresses.c.

References `con_addr()`, and `ip_subnet()`.

Referenced by `api_endpoint_auth()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_check_receive_quota()`, and `smtp_update_receive_stats()`.

5.155.2.17 uint32_t con_addr_word (connection_t * con, int_t position)

Get a specified 32-bit segment of a connection's peer IP address.

Parameters:

con a pointer to the connection object to be examined.

position a zero-based index into the 32-bit word(s) that comprise the target IP address.

Returns:

-1 if the connecting address can't be looked up, 0 on general error, or the specified 32-bit segment of the passed address on success.

Definition at line 134 of file addresses.c.

References `con_addr()`, and `ip_word()`.

5.155.2.18 uint64_t con_decrement_refs (connection_t * con)

[connections.c](#) [connections.c](#)

Parameters:

con the connection object being referenced.

Returns:

an integer containing the new number of references to the specified connection.

Definition at line 210 of file connections.c.

References `mutex_lock()`, and `mutex_unlock()`.

Referenced by `con_destroy()`.

5.155.2.19 void con_destroy (connection_t * con)

Destroy and free a generic connection object after executing its protocol-specific destructor; update any statistics accordingly.

Parameters:

con a pointer to the connection to be destroyed.

Returns:

This function returns no value.

Definition at line 104 of file connections.c.

References con_decrement_refs(), DMTP, dmtplib_session_destroy(), HTTP, http_session_destroy(), IMAP, imap_session_destroy(), mm_cleanup, mm_free(), MOLTEN, molten_session_destroy(), mutex_destroy(), POP, pop_session_destroy(), SMTP, smtp_session_destroy(), st_cleanup, stats_decrement_by_name(), SUBMISSION, and tls_free().

Referenced by con_reverse_lookup(), dmtplib_quit(), http_close(), imap_logout(), molten_quit(), pop_quit(), protocol_enqueue(), and smtp_quit().

5.155.2.20 void con_flush (connection_t * con)

Attempt to flush any buffered data associated with a network connection.

Parameters:

con the input client connection.

Returns:

This function returns no value.

Definition at line 89 of file connections.c.

References net_set_nodelay().

Referenced by smtp_quit().

5.155.2.21 uint64_t con_increment_refs (connection_t * con)

Increment a connection object's reference counter.

Parameters:

con the connection object being referenced.

Returns:

an integer containing the new number of references to the specified connection.

Definition at line 194 of file connections.c.

References mutex_lock(), and mutex_unlock().

Referenced by con_init().

5.155.2.22 connection_t* con_init (int cond, server_t * server)

Create a new connection object for a client connection.

Parameters:

cond the socket descriptor of the inbound client connection that was just accepted.

server the handle of the server that serviced the inbound request.

Definition at line 250 of file connections.c.

References `con_increment_refs()`, `connection_t`, `mm_alloc()`, `mm_free()`, `mutex_init()`, `server_t::network`, `server_t::sockd`, and `tcp_addr_ip()`.

Referenced by `protocol_process()`.

5.155.2.23 bool_t con_init_network_buffer (connection_t * con)

Allocate and initialize a new network buffer for a connection, if it is not already associated with one.

Note:

A network buffer of size `magma.system.network_buffer` bytes will be allocated for the connection if one does not exist.

Returns:

true if the connection object has been associated with a network buffer or false on failure.

Definition at line 226 of file connections.c.

References `log_info`, `magma`, `magma_t::network_buffer`, `pl_null()`, `st_alloc()`, and `magma_t::system`.

Referenced by `con_read()`, and `con_read_line()`.

5.155.2.24 bool_t con_localhost (connection_t * con)

Determine whether a client connection is from the same machine, using the loopback adapter, or a remote machine.

See also:

[ip_localhost\(\)](#)

Returns:

true if the connection appears to be using the loopback adapter, or false, if the connection appears to be from anywhere else.

Definition at line 32 of file connections.c.

References `ip_localhost()`.

Referenced by `json_api_dispatch()`, `portal_endpoint()`, `portal_endpoint_auth()`, `portal_process()`, and `portal_upload()`.

5.155.2.25 int64_t con_print (connection_t * con, chr_t * format, ...)

Write a formatted string to a network connection.

See also:

[con_write_bl\(\)](#)

Parameters:

con the connection across which the supplied data will be written.

format a format string for the output string to be written to the connection's remote client.

... a variable argument list of parameters for the specified format string.

Returns:

-1 on general failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 112 of file write.c.

References `buflen`, `bufptr`, `bytes`, `con_write_bl()`, `length`, `mm_alloc()`, and `mm_free()`.

Referenced by `dmtpl_ahlo()`, `dmtpl_helo()`, `dmtpl_init()`, `http_parse_header()`, `http_print_301()`, `http_response_header()`, `http_response_options()`, `imap_append()`, `imap_capability()`, `imap_check()`, `imap_close()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_id()`, `imap_idle()`, `imap_init()`, `imap_invalid()`, `imap_list()`, `imap_login()`, `imap_logout()`, `imap_lsub()`, `imap_noop()`, `imap_process()`, `imap_rename()`, `imap_search()`, `imap_select()`, `imap_starttls()`, `imap_status()`, `imap_store()`, `imap_subscribe()`, `imap_unsubscribe()`, `molten_stats()`, `pop_capa()`, `pop_last()`, `pop_list()`, `pop_retr()`, `pop_stat()`, `pop_top()`, `pop_uidl()`, `register_print_captcha()`, `smtp_data()`, `smtp_data_inbound()`, `smtp_data_outbound()`, `smtp_disabled()`, `smtp_ahlo()`, `smtp_helo()`, `smtp_init()`, `smtp_rcpt_to()`, and `teacher_print_message()`.

5.155.2.26 bool_t con_private (connection_t * con)

Determine whether a client connection is with a machine on the same private network.

See also:

[ip_private\(\)](#)

Returns:

true if the connection appears to be using the loopback adapter, or false, if the connection appears to be from anywhere else.

Definition at line 48 of file connections.c.

References `ip_localhost()`.

5.155.2.27 int64_t con_read (connection_t * con)

Read data from a network connection, and store the data in the connection context buffer.

Note:

This function handles reading data from both regular and SSL connections. If the connection's network buffer hasn't been allocated, it will be initialized.

Parameters:

con a pointer to the connection object from which the data will be read.

Returns:

-1 on general failure, -2 if the connection was reset, or the amount of data that was read.

Definition at line 108 of file read.c.

References `bytes`, `con_init_network_buffer()`, `con_status()`, `mm_move()`, `pl_length_get()`, `pl_null()`, `st_avail_get()`, `st_char_get()`, `st_data_get()`, `st_length_get()`, `st_length_set()`, `status`, `tcp_read()`, and `tls_read()`.

Referenced by `http_body()`, `imap_parse_literal()`, `smtp_data_finish()`, and `smtp_data_read()`.

5.155.2.28 int64_t con_read_line (connection_t * con, bool_t block)

Read a line of input from a network connection.

Note:

This function handles reading data from both regular and ssl connections. This function continually attempts to read incoming data from the specified connection until a terminated line of input is received. If a new line is read, the length of that line is returned to the caller, including the trailing . If the read returns -1 and wasn't caused by a syscall interruption or blocking error, -1 is returned, and the connection status is set to -1. If the read returns 0 and wasn't caused by a syscall interruption or blocking error, -2 is returned, and the connection status is set to 2. Once a character is reached, the length of the current line of input is returned to the user, and the connection status is set to 1.

Parameters:

con the network connection across which the line of data will be read.

Returns:

-1 on general failure, -2 if the connection was reset, or the length of the current line of input, including the trailing new line character.

Definition at line 21 of file read.c.

References bytes, con_init_network_buffer(), con_status(), line_pl_st(), mm_move(), pl_empty(), pl_length_get(), pl_null(), st_avail_get(), st_char_get(), st_data_get(), st_empty, st_length_get(), st_length_set(), status, tcp_read(), and tls_read().

Referenced by dmtcp_process(), http_process(), imap_parse_literal(), imap_process(), molten_parse(), pop_process(), smtp_auth_login(), smtp_auth_plain(), and smtp_process().

5.155.2.29 stringer_t* con_reverse_check (connection_t * con, uint32_t timeout)

[reverse.c](#) [reverse.c](#)

Note:

Polling will occur every 100ms, for as many seconds as specified by the user.

Parameters:

con the specified connection object to be polled that is the target of the DNS lookup.

timeout the number of seconds to wait for the lookup operation to complete.

Returns:

NULL on failure, or a pointer to a managed string containing the connection's peer hostname on success.

Definition at line 75 of file reverse.c.

References mutex_lock(), mutex_unlock(), REVERSE_COMPLETE, REVERSE_PENDING, and status.

Referenced by mail_add_inbound_headers(), and mail_add_outbound_headers().

5.155.2.30 void con_reverse_domain (connection_t * con, stringer_t * domain, int_t status)

Set the domain name and reverse lookup status of a connection.

Note:

Possible values for status include REVERSE_ERROR, REVERSE_EMPTY, REVERSE_PENDING, and REVERSE_COMPLETE.

Parameters:

domain the new value of the connection's hostname.

status the new value of the connection's reverse lookup status.

Returns:

This function returns no value.

Definition at line 42 of file reverse.c.

References mutex_lock(), and mutex_unlock().

Referenced by con_reverse_lookup().

5.155.2.31 void con_reverse_enqueue (connection_t * con)

Queue a reverse DNS lookup on the specified connection, if one hasn't been performed.

Parameters:

con the connection object to be examined.

Returns:

This function returns no value.

Definition at line 15 of file reverse.c.

References con_reverse_lookup(), enqueue(), mutex_lock(), mutex_unlock(), REVERSE_EMPTY, and REVERSE_PENDING.

Referenced by dmtip_init(), http_init(), imap_init(), and smtp_init().

5.155.2.32 void con_reverse_lookup (connection_t * con)

Perform a reverse DNS lookup on the remote end of a connection, and save the hostname.

Parameters:

con the connection object to be queried.

Returns:

This function returns no value.

Definition at line 107 of file reverse.c.

References con_destroy(), con_reverse_domain(), con_reverse_status(), MEMORYBUF, ns_length_get(), REVERSE_COMPLETE, REVERSE_ERROR, and st_import().

Referenced by con_reverse_enqueue().

5.155.2.33 void con_reverse_status (connection_t * con, int_t status)

Definition at line 59 of file reverse.c.

References mutex_lock(), and mutex_unlock().

Referenced by con_reverse_lookup().

5.155.2.34 int_t con_secure (connection_t * con)

Return the security level of a specified connection.

Parameters:

con the input client connection.

Returns:

1 for a secure connection, 0 for insecure, and -1 if the server does not support SSL/TLS.

Definition at line 15 of file connections.c.

Referenced by `contact_process()`, `http_print_301()`, `http_response_cookie()`, `imap_capability()`, `imap_init()`, `imap_login()`, `imap_starttls()`, `json_api_dispatch()`, `pop_capa()`, `pop_pass()`, `pop_starttls()`, `portal_endpoint()`, `portal_endpoint_auth()`, `portal_process()`, `portal_upload()`, `protocol_enqueue()`, `register_process()`, `sess_create()`, `sess_get()`, `smtp_ehlo()`, `smtp_rcpt_to()`, `smtp_starttls()`, and `teacher_process()`.

5.155.2.35 int_t con_status (connection_t * con)

Return the status of a specified connection.

Parameters:

con the input client connection.

Returns:

-1 on network errors, 0 for an unknown status, 1 for connected, and 2 for a graceful shutdown.

Definition at line 64 of file connections.c.

References `tcp_status()`, and `tls_status()`.

Referenced by `con_read()`, `con_read_line()`, `con_write_bl()`, `dmtip_quit()`, `dmtip_requeue()`, `http_requeue()`, `imap_fetch()`, `imap_logout()`, `imap_requeue()`, `pop_quit()`, `pop_requeue()`, `smtp_quit()`, and `smtp_requeue()`.

5.155.2.36 int64_t con_write_bl (connection_t * con, char * block, size_t length)

Write data to a network connection. **HIGH:** Create a simpler method of triggering a queue event following a connection write operation, and audit the code to ensure write calls do not accidentally orphan a connection by not queuing the connection upon completion. In other words, always ensure that `enqueue()` is being called on a connection after all processing is performed, so that it is not lost (whether it is to be kept or not).

Note:

This function works regardless of whether or not the connection is ssl-enabled. If the network write requires multiple system calls, then this code will loop until all the data has been transmitted.

Parameters:

con the connection across which the supplied data will be written.

block a pointer to a data buffer containing the data to be written to the connection's remote client.

length the length, in bytes, of the data buffer to be written.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 24 of file write.c.

References `bytes`, `con_status()`, `status`, `tcp_write()`, and `tls_write()`.

Referenced by `con_print()`, `con_write_ns()`, `con_write_pl()`, `con_write_st()`, `dmtip_data()`, `dmtip_help()`, `dmtip_hist()`, `dmtip_invalid()`, `dmtip_mail()`, `dmtip_mode()`, `dmtip_noop()`, `dmtip_quit()`, `dmtip_rcpt()`, `dmtip_rset()`, `dmtip_sgnt()`, `dmtip_verb()`, `dmtip_vrfy()`, `imap_fetch()`, `imap_logout()`, `imap_parse_literal()`, `imap_process()`, `molten_invalid()`, `molten_stats()`, `pop_delete()`, `pop_init()`, `pop_invalid()`, `pop_list()`, `pop_noop()`, `pop_pass()`, `pop_quit()`, `pop_retr()`, `pop_rset()`, `pop_starttls()`, `pop_top()`, `pop_uidl()`, `pop_user()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_data()`, `smtp_data_inbound()`, `smtp_data_outbound()`, `smtp_ehlo()`, `smtp_helo()`, `smtp_invalid()`, `smtp_mail_from()`, `smtp_noop()`, `smtp_quit()`, `smtp_rcpt_to()`, `smtp_rset()`, and `smtp_starttls()`.

5.155.2.37 int64_t con_write_ns (connection_t * con, char * string)

Write a null-terminated string to a network connection.

See also:

[con_write_bl\(\)](#)

Parameters:

con the connection across which the supplied data will be written.

string a pointer to a null-terminated string containing the data to be written to the connection's remote client.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 89 of file write.c.

References [con_write_bl\(\)](#), and [ns_length_get\(\)](#).

5.155.2.38 int64_t con_write_pl (connection_t * con, placer_t string)

Write a placer to a network connection.

See also:

[con_write_bl\(\)](#)

Parameters:

con the connection across which the supplied data will be written.

string a placer pointing to the data to be written to the connection's remote client.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 100 of file write.c.

References [con_write_bl\(\)](#), [pl_char_get\(\)](#), and [pl_length_get\(\)](#).

5.155.2.39 int64_t con_write_st (connection_t * con, stringer_t * string)

Write a managed string to a network connection.

See also:

[con_write_bl\(\)](#)

Parameters:

con the connection across which the supplied data will be written.

string a managed string containing the data to be written to the connection's remote client.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 78 of file write.c.

References `con_write_bl()`, `st_char_get()`, and `st_length_get()`.

Referenced by `api_response()`, `contact_print_form()`, `contact_print_message()`, `http_print_400()`, `http_print_403()`, `http_print_404()`, `http_print_405()`, `http_print_500()`, `http_print_500_log()`, `http_print_501()`, `http_response()`, `imap_fetch()`, `imap_search()`, `pop_retr()`, `pop_top()`, `portal_endpoint_attachments_progress()`, `portal_endpoint_error()`, `portal_endpoint_response()`, `portal_endpoint_search()`, `portal_print_login()`, `register_print_captcha()`, `register_print_message()`, `register_print_step1()`, `register_print_step2()`, `register_print_step3()`, `smtp_data_outbound()`, `statistics_process()`, `teacher_print_form()`, and `teacher_print_message()`.

5.155.2.40 `bool_t net_init (server_t * server)`

[listeners.c](#)

Definition at line 58 of file listeners.c.

References `server_t::ipv6`, `server_t::listen_queue`, `log_critical`, `magma`, `mm_wipe()`, `net_set_buffer_length()`, `net_set_reuseable_address()`, `server_t::network`, `magma_t::network_buffer`, `server_t::port`, `server_t::sockd`, and `magma_t::system`.

Referenced by `servers_network_start()`.

5.155.2.41 `void net_listen (void)`

Definition at line 33 of file listeners.c.

References `server_t::enabled`, `magma`, `MAGMA_SERVER_INSTANCES`, `mm_free()`, `mm_wipe()`, `net_accept()`, `server_t::network`, `magma_t::servers`, `server_t::sockd`, `thread_alloc()`, and `thread_join()`.

Referenced by `main()`.

5.155.2.42 `bool_t net_set_blocking (int sd, bool_t blocking)`

Set the blocking flag for a socket.

Parameters:

sd the socket descriptor to be adjusted.

blocking a boolean, where true indicates the socket should be configured to use blocking system calls, and false to indicate the socket should use non-blocking calls.

Returns:

true if the flag was successfully set or false on failure.

Definition at line 103 of file options.c.

References `buflen`, `bufptr`, `log_error`, and `log_pedantic`.

5.155.2.43 `bool_t net_set_buffer_length (int sd, int buffer_recv, int buffer_send)`

[options.c](#) [options.c](#)

Parameters:

sd the socket descriptor to be configured.

buffer_recv the length, in bytes, of the socket receive buffer.

buffer_send the length, in bytes, of the socket send buffer.

Returns:

true if both buffer values were set successfully, or false on failure.

Definition at line 163 of file options.c.

References buflen, bufptr, and log_pedantic.

Referenced by net_init().

5.155.2.44 bool_t net_set_keepalive (int sd, bool_t keepalive, int_t idle, int_t interval, int_t tolerance)

Set the keepalive flag, the. Controls whether to send periodic messages along idle socket connections to ensure the connected peer is still present. If the peer fails to respond the connection can be broken.

Parameters:

sd The socket being configured.

keepalive Whether to enable the keepalive process.

idle Controls how long a connection must be idle (in seconds) before the system will start sending keep alive probes.

interval They delay (in seconds) between each keep alive probe.

tolerance The maximum number of failed probes that must be sent before dropping a connection.

Returns:

Returns true if all the parameters are configured properly, otherwise false to indicate an error.

Definition at line 23 of file options.c.

References buflen, bufptr, and log_pedantic.

5.155.2.45 bool_t net_set_linger (int sd, bool_t linger, int_t timeout)

Set the linger flag for a socket.

Parameters:

sd the socket descriptor to be adjusted.

linger a boolean variable specifying whether the socket will linger before closing until all queued messages for the socket have been sent.

timeout a value specifying the number of seconds to linger, if linger is true.

Returns:

true if the flag was successfully set or false on failure.

Definition at line 63 of file options.c.

References buflen, bufptr, and log_pedantic.

5.155.2.46 bool_t net_set_nodelay (int sd, bool_t nodelay)

Set the delay flag for a socket (to disable the Nagle algorithm).

Parameters:

sd the socket descriptor to be adjusted.

nodelay a boolean variable specifying whether the Nagle algorithm should be disabled (true) or not (false).

Returns:

true if the flag was successfully set or false on failure.

Definition at line 44 of file options.c.

References `buflen`, `bufptr`, and `log_pedantic`.

Referenced by `con_flush()`.

5.155.2.47 `bool_t net_set_reuseable_address (int sd, bool_t reuse)`

Set the reusable flag for a socket.

Parameters:

sd the socket descriptor to be adjusted.

reuse a boolean variable specifying whether the listening socket should be reusable or not.

Returns:

true if the flag was successfully set or false on failure.

Definition at line 84 of file options.c.

References `buflen`, `bufptr`, and `log_pedantic`.

Referenced by `net_init()`.

5.155.2.48 `bool_t net_set_timeout (int sd, uint32_t timeout_rcv, uint32_t timeout_send)`

Set the send and receive timeouts for a socket.

Parameters:

sd the socket descriptor to be configured.

timeout_rcv the receive timeout value for the socket, in seconds.

timeout_send the send timeout value for the socket, in seconds.

Returns:

true if both timeout values were set successfully, or false on failure.

Definition at line 131 of file options.c.

References `buflen`, `bufptr`, `log_pedantic`, and `mm_wipe()`.

Referenced by `protocol_process()`, and `smtp_client_connect()`.

5.155.2.49 `void net_shutdown (server_t * server)`

Close the listening socket associated with a server.

Returns:

This function returns no value.

Definition at line 216 of file listeners.c.

References `server_t::network`, and `server_t::sockd`.

Referenced by `servers_network_stop()`.

5.155.2.50 `void net_trigger (bool_t verbose)`

Triggers a connection for each server instance, allowing the the listeners to shutdown cleanly. And then purges any remaining sockets.

LOW: This only compares the port number for the sockets. We should also ensure the socket is owned by magmad, and/or that the server used INADDR_ANY or IN6ADDR_ANY_INIT, otherwise the logic below will close sockets that could be owned by other processes on the system that are using the same port number, while bound to a different IP interface.

Definition at line 129 of file listeners.c.

References `client_close()`, `client_connect()`, `client_t`, `server_t::enabled`, `ip_t::family`, `ip_t::ip4`, `ip_t::ip6`, `ip_presentation()`, `log_critical`, `log_info`, `magma`, `MAGMA_SERVER_INSTANCES`, `MEMORYBUF`, `mm_copy()`, `mm_wipe()`, `server_t::network`, `PLACER`, `server_t::port`, `queue_signal()`, `magma_t::servers`, `servers_get_count_using_port()`, `server_t::sockd`, and `st_char_get()`.

Referenced by `signal_shutdown()`.

5.155.2.51 `stringer_t* protocol_type (connection_t * con)`

Definition at line 11 of file protocol.c.

References `CONSTANT`, `DMTP`, `HTTP`, `IMAP`, `log_check`, `MOLTEN`, `POP`, `SMTP`, `SUBMISSION`, and `TLS_PORT`.

Referenced by `protocol_secure()`.

5.155.3 Variable Documentation

5.155.3.1 `client_t`

Definition at line 41 of file network.h.

Referenced by `client_connect()`, `net_trigger()`, `portal_smtp_relay_message()`, `smtp_bounce()`, `smtp_client_connect()`, `smtp_forward_message()`, `smtp_relay_message()`, `smtp_reply()`, and `smtp_send_message()`.

5.155.3.2 `command_t`

Definition at line 30 of file network.h.

Referenced by `__attribute__()`, `dmtp_compare()`, `dmtp_process()`, `dmtp_sort()`, `imap_compare()`, `imap_process()`, `imap_sort()`, `molten_compare()`, `molten_parse()`, `molten_sort()`, `pop_compare()`, `pop_process()`, `pop_sort()`, `portal_endpoint()`, `portal_endpoint_compare()`, `portal_endpoint_sort()`, `smtp_compare()`, `smtp_process()`, and `smtp_sort()`.

5.155.3.3 `connection_t`

Definition at line 75 of file network.h.

Referenced by `con_init()`, `portal_endpoint()`, and `protocol_process()`.

5.156 src/providers/dime/common/network.h File Reference

```
#include <string.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include "dime/common/error.h"
```

Functions

- **PUBLIC_FUNC_DECL** (int, connect_host, const char *hostname, unsigned short port, int force_family)
- int **_connect_timeout** (int fd, const struct sockaddr *addr, socklen_t addrlen)

Attempt a TCP connection with a time-out mechanism.

5.156.1 Function Documentation

5.156.1.1 int _connect_timeout (int fd, struct sockaddr const * addr, socklen_t addrlen)

Attempt a TCP connection with a time-out mechanism.

Parameters:

- fd** the open socket descriptor across which the connection will be attempted.
- addr** a pointer to the prepared sockaddr structure that is the target of the connection.
- addrlen** the size of the supplied sockaddr structure.

Returns:

- 1 on general error, 0 if the connection failed or timed out, and 1 on success.

Definition at line 113 of file network.c.

References CONNECT_TIMEOUT, ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_INT.

Referenced by _connect_host().

5.156.1.2 PUBLIC_FUNC_DECL (int, connect_host, const char * hostname, unsigned short port, int force_family)

5.157 src/network/options.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t net_set_keepalive](#) (int sd, [bool_t](#) keepalive, [int_t](#) idle, [int_t](#) interval, [int_t](#) tolerance)
Set the keepalive flag, the.
- [bool_t net_set_nodelay](#) (int sd, [bool_t](#) nodelay)
Set the delay flag for a socket (to disable the Nagle algorithm).
- [bool_t net_set_linger](#) (int sd, [bool_t](#) linger, [int_t](#) timeout)
Set the linger flag for a socket.
- [bool_t net_set_reuseable_address](#) (int sd, [bool_t](#) reuse)
Set the reusable flag for a socket.
- [bool_t net_set_blocking](#) (int sd, [bool_t](#) blocking)
Set the blocking flag for a socket.
- [bool_t net_set_timeout](#) (int sd, uint32_t timeout_recv, uint32_t timeout_send)
Set the send and receive timeouts for a socket.
- [bool_t net_set_buffer_length](#) (int sd, int buffer_recv, int buffer_send)
Set the send and receive buffers for a socket at the system level.

5.157.1 Function Documentation

5.157.1.1 bool_t net_set_blocking (int sd, bool_t blocking)

Set the blocking flag for a socket.

Parameters:

sd the socket descriptor to be adjusted.

blocking a boolean, where true indicates the socket should be configured to use blocking system calls, and false to indicate the socket should use non-blocking calls.

Returns:

true if the flag was successfully set or false on failure.

Definition at line 103 of file options.c.

References [buflen](#), [bufptr](#), [log_error](#), and [log_pedantic](#).

5.157.1.2 bool_t net_set_buffer_length (int sd, int buffer_recv, int buffer_send)

Set the send and receive buffers for a socket at the system level. [options.c](#)

Parameters:

sd the socket descriptor to be configured.

buffer_recv the length, in bytes, of the socket receive buffer.

buffer_send the length, in bytes, of the socket send buffer.

Returns:

true if both buffer values were set successfully, or false on failure.

Definition at line 163 of file options.c.

References buflen, bufptr, and log_pedantic.

Referenced by net_init().

5.157.1.3 bool_t net_set_keepalive (int sd, bool_t keepalive, int_t idle, int_t interval, int_t tolerance)

Set the keepalive flag, the. Controls whether to send periodic messages along idle socket connections to ensure the connected peer is still present. If the peer fails to respond the connection can be broken.

Parameters:

sd The socket being configured.

keepalive Whether to enable the keepalive process.

idle Controls how long a connection must be idle (in seconds) before the system will start sending keep alive probes.

interval They delay (in seconds) between each keep alive probe.

tolerance The maximum number of failed probes that must be sent before dropping a connection.

Returns:

Returns true if all the parameters are configured properly, otherwise false to indicate an error.

Definition at line 23 of file options.c.

References buflen, bufptr, and log_pedantic.

5.157.1.4 bool_t net_set_linger (int sd, bool_t linger, int_t timeout)

Set the linger flag for a socket.

Parameters:

sd the socket descriptor to be adjusted.

linger a boolean variable specifying whether the socket will linger before closing until all queued messages for the socket have been sent.

timeout a value specifying the number of seconds to linger, if linger is true.

Returns:

true if the flag was successfully set or false on failure.

Definition at line 63 of file options.c.

References buflen, bufptr, and log_pedantic.

5.157.1.5 **bool_t net_set_nodelay (int *sd*, bool_t *nodelay*)**

Set the delay flag for a socket (to disable the Nagle algorithm).

Parameters:

sd the socket descriptor to be adjusted.

nodelay a boolean variable specifying whether the Nagle algorithm should be disabled (true) or not (false).

Returns:

true if the flag was successfully set or false on failure.

Definition at line 44 of file options.c.

References buflen, bufptr, and log_pedantic.

Referenced by con_flush().

5.157.1.6 **bool_t net_set_reuseable_address (int *sd*, bool_t *reuse*)**

Set the reusable flag for a socket.

Parameters:

sd the socket descriptor to be adjusted.

reuse a boolean variable specifying whether the listening socket should be reusable or not.

Returns:

true if the flag was successfully set or false on failure.

Definition at line 84 of file options.c.

References buflen, bufptr, and log_pedantic.

Referenced by net_init().

5.157.1.7 **bool_t net_set_timeout (int *sd*, uint32_t *timeout_rcv*, uint32_t *timeout_snd*)**

Set the send and receive timeouts for a socket.

Parameters:

sd the socket descriptor to be configured.

timeout_rcv the receive timeout value for the socket, in seconds.

timeout_snd the send timeout value for the socket, in seconds.

Returns:

true if both timeout values were set successfully, or false on failure.

Definition at line 131 of file options.c.

References buflen, bufptr, log_pedantic, and mm_wipe().

Referenced by protocol_process(), and smtp_client_connect().

5.158 src/network/pop.h File Reference

Data Structures

- struct [__attribute__](#)

5.159 src/servers/pop/pop.h File Reference

Functions

- `int_t pop_compare` (`const void *compare`, `const void *command`)
commands.c
- `void pop_process` (`connection_t *con`)
- `void pop_requeue` (`connection_t *con`)
- `void pop_sort` (`void`)
Sort the POP3 command table to be ready for binary searches.
- `uint64_t pop_get_last` (`inx_t *messages`)
mailbox.c
- `meta_message_t * pop_get_message` (`inx_t *messages`, `uint64_t get`)
Get a message by its pop sequence number.
- `uint64_t pop_total_messages` (`inx_t *messages`)
Get the number of messages available to a POP3 user.
- `uint64_t pop_total_size` (`inx_t *messages`)
Get the total size of all messages available to a POP3 user.
- `bool_t pop_num_parse` (`connection_t *con`, `uint64_t *outnum`, `bool_t required`)
parse.c
- `stringer_t * pop_pass_parse` (`connection_t *con`)
Parse a POP3 PASS command.
- `bool_t pop_top_parse` (`connection_t *con`, `uint64_t *number`, `uint64_t *lines`)
Parse the arguments to a POP3 TOP command.
- `stringer_t * pop_user_parse` (`connection_t *con`)
Parse a POP3 USER command.
- `void pop_capa` (`connection_t *con`)
pop.c
- `void pop_dele` (`connection_t *con`)
Get a message, in response to a POP3 DELE command.
- `void pop_init` (`connection_t *con`)
Initialize a new POP3 connection.
- `void pop_invalid` (`connection_t *con`)
A function handler for invalid POP3 commands.
- `void pop_last` (`connection_t *con`)
Get the sequence number of the last read message, in response to a POP3 LAST command.
- `void pop_list` (`connection_t *con`)
Get the list of a user's messages, in response to a POP3 LIST command.

- void `pop_noop` (`connection_t *con`)
Execute a POP3 no-operation command.
- void `pop_pass` (`connection_t *con`)
Accept and verify a password for POP3 authentication.
- void `pop_quit` (`connection_t *con`)
Gracefully destroy a POP3 session, whether because of an error or in response to a user QUIT command.
- void `pop_retr` (`connection_t *con`)
Retrieve a user's message, in response to a POP3 RETR command.
- void `pop_rset` (`connection_t *con`)
Reset the user's mailbox, in response to a POP3 RSET command.
- void `pop_starttls` (`connection_t *con`)
TODO: Review error messages and update them with the appropriate response code, as per RFC 3206 regarding the response codes extension.
- void `pop_stat` (`connection_t *con`)
Display a user's message statistics, in response to a POP3 STAT command.
- void `pop_top` (`connection_t *con`)
Get the top lines of a message or collection of messages, in response to a POP3 TOP command.
- void `pop_uidl` (`connection_t *con`)
Get the UIDL for a message or collection of messages, in response to a POP3 UIDL command.
- void `pop_user` (`connection_t *con`)
Accept a username for POP3 authentication.
- void `pop_session_destroy` (`connection_t *con`)
sessions.c
- `int_t pop_session_reset` (`connection_t *con`)
Reset a POP3 session by guaranteeing that no messages are flagged to be deleted.

5.159.1 Function Documentation

5.159.1.1 void `pop_capa` (`connection_t *con`)

`pop.c pop.c`

Note:

See RFC 2449 POP3 Extension Mechanism for details.

Parameters:

`con` the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 342 of file pop.c.

References build_version(), con_print(), and con_secure().

5.159.1.2 int_t pop_compare (const void * *compare*, const void * *command*)

commands.c

Definition at line 11 of file commands.c.

References command_t, PLACER, st_cmp_ci_eq(), and st_cmp_ci_starts().

Referenced by pop_process(), and pop_sort().

5.159.1.3 void pop_dele (connection_t * *con*)

Get a message, in response to a POP3 DELE command.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 478 of file pop.c.

References con_write_bl(), MAIL_STATUS_HIDDEN, meta_message_t, meta_user_unlock(), meta_user_wlock(), number, pop_get_message(), pop_invalid(), and pop_num_parse().

5.159.1.4 uint64_t pop_get_last (inx_t * *messages*)

[mailbox.c](#) [mailbox.c](#)

Note:

This function only counts messages that aren't deleted or hidden, and weren't created by the IMAP APPEND command.

Parameters:

messages an inx holder containing a collection of messages to be analyzed.

Returns:

the sequence number of the last message that isn't flagged as recent, or 0 on failure.

Definition at line 74 of file mailbox.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_APPENDED, MAIL_STATUS_HIDDEN, MAIL_STATUS_RECENT, and meta_message_t.

Referenced by pop_last().

5.159.1.5 meta_message_t* pop_get_message (inx_t * *messages*, uint64_t *get*)

Get a message by its pop sequence number.

Parameters:

messages an inx holder containing the collection of the user's messages to be traversed.

number the zero-based pop sequence number of the message to be retrieved.

Returns:

NULL on failure or the meta message object of the message if it was found.

Definition at line 113 of file mailbox.c.

References `count`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `MAIL_STATUS_APPENDED`, and `meta_message_t`.

Referenced by `pop_dele()`, `pop_list()`, `pop_retr()`, `pop_top()`, and `pop_uidl()`.

5.159.1.6 void pop_init (connection_t * con)

Initialize a new POP3 connection.

Parameters:

con the newly connected POP3 client connection.

Returns:

This function returns no value.

Definition at line 723 of file pop.c.

References `con_write_bl()`, and `pop_requeue()`.

Referenced by `protocol_enqueue()`.

5.159.1.7 void pop_invalid (connection_t * con)

A function handler for invalid POP3 commands.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 67 of file pop.c.

References `con_write_bl()`.

Referenced by `pop_dele()`, `pop_last()`, `pop_list()`, `pop_pass()`, `pop_process()`, `pop_retr()`, `pop_rset()`, `pop_starttls()`, `pop_stat()`, `pop_top()`, `pop_uidl()`, and `pop_user()`.

5.159.1.8 void pop_last (connection_t * con)

Get the sequence number of the last read message, in response to a POP3 LAST command.

See also:

[pop_get_last\(\)](#)

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 390 of file pop.c.

References `con_print()`, `meta_user_rlock()`, `meta_user_unlock()`, `number`, `pop_get_last()`, and `pop_invalid()`.

5.159.1.9 void pop_list (connection_t * con)

Get the list of a user's messages, in response to a POP3 LIST command.

See also:

[pop_total_messages\(\)](#), [pop_get_message\(\)](#)

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 415 of file pop.c.

References `con_print()`, `con_write_bl()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `MAIL_STATUS_APPENDED`, `MAIL_STATUS_HIDDEN`, `meta_message_t`, `meta_user_rlock()`, `meta_user_unlock()`, `number`, `pop_get_message()`, `pop_invalid()`, `pop_num_parse()`, and `pop_total_messages()`.

5.159.1.10 void pop_noop (connection_t * con)

Execute a POP3 no-operation command.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 56 of file pop.c.

References `con_write_bl()`.

5.159.1.11 bool_t pop_num_parse (connection_t * con, uint64_t * outnum, bool_t required)

parse.c parse.c

Parameters:

con the connection across which the pop command was issued.

outnum a pointer to an unsigned 64-bit integer to receive the numerical argument of the pop command on success.

required specifies whether or not a parameter is required.

Returns:

true if the pop command contained a valid unsigned number or false on failure.

Definition at line 17 of file parse.c.

References `length`, `log_pedantic`, `st_data_get()`, `st_length_get()`, and `uint64_conv_bl()`.

Referenced by `pop_dele()`, `pop_list()`, `pop_retr()`, and `pop_uidl()`.

5.159.1.12 void pop_pass (connection_t * con)

Accept and verify a password for POP3 authentication.

Note:

This command is only allowed for sessions which have not yet been authenticated, but which have already supplied a username. If the username/password combo was validated, the account information is retrieved and checked to see if it is locked. After successful authentication, this function will prohibit insecure connections for any user configured to use TLS only, and enforce the existence of only one POP3 session at a time. Finally, the database Log table for this user's POP3 access is updated, and all the user's messages are retrieved.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 168 of file pop.c.

References `auth_free()`, `AUTH_LOCK_ABUSE`, `AUTH_LOCK_ADMIN`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_USER`, `auth_login()`, `cache_decrement()`, `cache_increment()`, `con_addr_presentation()`, `con_addr_subnet()`, `con_secure()`, `con_write_bl()`, `inx_count()`, `auth_t::keys`, `lock_get()`, `lock_release()`, `auth_t::locked`, `log_info`, `MANAGEDBUF`, `auth_t::master`, `meta_data_update_log()`, `meta_get()`, `META_GET_KEYS`, `META_GET_MESSAGES`, `meta_inx_remove()`, `META_LOCKED`, `meta_messages_login_update()`, `META_PROTOCOL_POP`, `META_USER_ENCRYPT_DATA`, `meta_user_ref_protocol_total()`, `META_USER_TLS`, `meta_user_unlock()`, `meta_user_wlock()`, `pop_invalid()`, `pop_pass_parse()`, `auth_t::salt`, `auth_t::seasoning`, `st_char_get()`, `st_dupe()`, `st_empty`, `st_free()`, `st_length_int()`, `st_populated`, `st_quick()`, `auth_t::status`, `time_datestamp()`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, and `auth_t::verification`.

5.159.1.13 stringer_t* pop_pass_parse (connection_t * con)

Parse a POP3 PASS command.

Note:

This function will stop reading in password characters when an invalid character is encountered.

Parameters:

con the POP3 client connection that issued the PASS command.

Returns:

a managed string containing the user supplied password, or NULL on failure.

Definition at line 191 of file parse.c.

References `CONTIGUOUS`, `length`, `log_error`, `log_pedantic`, `MANAGED_T`, `SECURE`, `st_alloc_opts()`, `st_copy_in()`, `st_data_get()`, and `st_length_get()`.

Referenced by `pop_pass()`.

5.159.1.14 void pop_process (connection_t * con)

Definition at line 43 of file commands.c.

References command, command_t, con_read_line(), enqueue(), pl_char_get(), pl_empty(), pl_length_get(), pop_commands, pop_compare(), pop_invalid(), pop_process(), pop_quit(), pop_requeue(), and requeue().

Referenced by pop_process(), and pop_requeue().

5.159.1.15 void pop_quit (connection_t * con)

Gracefully destroy a POP3 session, whether because of an error or in response to a user QUIT command.

Parameters:

con the POP3 client connection to be shut down. This function returns no value.

Definition at line 107 of file pop.c.

References con_destroy(), con_status(), and con_write_bl().

Referenced by pop_process(), and pop_requeue().

5.159.1.16 void pop_requeue (connection_t * con)

Definition at line 31 of file commands.c.

References con_status(), enqueue(), pop_process(), pop_quit(), and status.

Referenced by pop_init(), and pop_process().

5.159.1.17 void pop_retr (connection_t * con)

Retrieve a user's message, in response to a POP3 RETR command.

Note:

This function will fail if a deleted message was specified by the user.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 653 of file pop.c.

References con_print(), con_write_bl(), con_write_st(), mail_destroy(), mail_load_message(), MAIL_STATUS_HIDDEN, meta_message_t, meta_user_rlock(), meta_user_unlock(), number, PLACER, pop_get_message(), pop_invalid(), pop_num_parse(), st_char_get(), st_length_get(), st_replace(), and mail_message_t::text.

5.159.1.18 void pop_rset (connection_t * con)

Reset the user's mailbox, in response to a POP3 RSET command.

See also:

[pop_session_reset\(\)](#)

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 82 of file pop.c.

References `con_write_bl()`, `pop_invalid()`, and `pop_session_reset()`.

5.159.1.19 void pop_session_destroy (connection_t * con)

sessions.c sessions.c

Parameters:

con the POP3 client connection issuing the command. This function returns no value.

Definition at line 50 of file sessions.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_delete()`, `M_TYPE_UINT64`, `mail_cache_reset()`, `mail_remove_message()`, `MAIL_STATUS_HIDDEN`, `meta_inx_remove()`, `meta_message_t`, `meta_messages_update_sequences()`, `META_PROTOCOL_POP`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_MESSAGES`, `serial_increment()`, `st_cleanup`, `multi_t::u64`, and `multi_t::val`.

Referenced by `con_destroy()`.

5.159.1.20 int_t pop_session_reset (connection_t * con)

Reset a POP3 session by guaranteeing that no messages are flagged to be deleted.

Parameters:

con the POP3 client connection issuing the command.

Returns:

-1 on general error, 0 if the connection hasn't been authenticated, or 1 if all messages have had their statuses successfully reset.

Definition at line 15 of file sessions.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `MAIL_STATUS_HIDDEN`, `meta_message_t`, `meta_user_unlock()`, and `meta_user_wlock()`.

Referenced by `pop_rset()`, and `pop_starttls()`.

5.159.1.21 void pop_sort (void)

Sort the POP3 command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 26 of file commands.c.

References `command_t`, `pop_commands`, and `pop_compare()`.

Referenced by `protocol_init()`.

5.159.1.22 void pop_starttls (connection_t * con)

TODO: Review error messages and update them with the appropriate response code, as per RFC 3206 regarding the response codes extension. Initialize a TLS session for an unauthenticated POP3 session.

Note:

RFC 2595 / section 4 dictates that the STLS/STARTTLS command should only be available in the authorization state.

Parameters:

con the connection of the POP3 client requesting the transport layer security upgrade.

Returns:

This function returns no value (all error messages are written directly to the requesting client).

Definition at line 18 of file pop.c.

References con_secure(), con_write_bl(), log_pedantic, M_SSL_BIO_NOCLOSE, pl_null(), pop_invalid(), pop_session_reset(), st_length_set(), stats_increment_by_name(), and tls_server_alloc().

5.159.1.23 void pop_stat (connection_t * con)

Display a user's message statistics, in response to a POP3 STAT command.

See also:

[pop_total_messages\(\)](#), [pop_total_size\(\)](#)

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 364 of file pop.c.

References con_print(), count, meta_user_rlock(), meta_user_unlock(), pop_invalid(), pop_total_messages(), and pop_total_size().

5.159.1.24 void pop_top (connection_t * con)

Get the top lines of a message or collection of messages, in response to a POP3 TOP command.

Note:

This function will fail if a deleted message was specified by the user.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 582 of file pop.c.

References con_print(), con_write_bl(), con_write_st(), mail_destroy(), mail_load_message_top(), MAIL_STATUS_HIDDEN, meta_message_t, meta_user_rlock(), meta_user_unlock(), number, PLACER, pop_get_message(), pop_invalid(), pop_top_parse(), st_char_get(), st_length_get(), st_replace(), and mail_message_t::text.

5.159.1.25 bool_t pop_top_parse (connection_t * con, uint64_t * number, uint64_t * lines)

Parse the arguments to a POP3 TOP command.

Parameters:

con the POP3 client connection that issued the TOP command.

number a pointer to an unsigned 64 bit integer to receive the value of the TOP command's message sequence number.

lines a pointer to an unsigned 64 bit integer to receive the value of the TOP command's line number count.

Returns:

true on success or false if an error was encountered.

Definition at line 89 of file parse.c.

References length, log_pedantic, st_data_get(), st_length_get(), and uint64_conv_bl().

Referenced by pop_top().

5.159.1.26 uint64_t pop_total_messages (inx_t * messages)

Get the number of messages available to a POP3 user.

Note:

This function only counts messages that aren't deleted or hidden, and weren't created by the IMAP APPEND command.

Parameters:

messages an inx holder containing a collection of messages to be analyzed.

Returns:

the total number of messages available to the POP3 user, or 0 on failure.

Definition at line 16 of file mailbox.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_APPENDED, MAIL_STATUS_HIDDEN, and meta_message_t.

Referenced by pop_list(), pop_stat(), and pop_uidl().

5.159.1.27 uint64_t pop_total_size (inx_t * messages)

Get the total size of all messages available to a POP3 user.

Note:

This function only counts messages that aren't deleted or hidden, and weren't created by the IMAP APPEND command.

Parameters:

messages an inx holder containing a collection of messages to be analyzed.

Returns:

the total size, in bytes, of all messages available to the POP3 user, or 0 on failure.

Definition at line 45 of file mailbox.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_APPENDED, MAIL_STATUS_HIDDEN, and meta_message_t.

Referenced by pop_stat().

5.159.1.28 void pop_uidl (connection_t * con)

Get the UIDL for a message or collection of messages, in response to a POP3 UIDL command.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 517 of file pop.c.

References con_print(), con_write_bl(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_APPENDED, MAIL_STATUS_HIDDEN, meta_message_t, meta_user_rlock(), meta_user_unlock(), number, pop_get_message(), pop_invalid(), pop_num_parse(), and pop_total_messages().

5.159.1.29 void pop_user (connection_t * con)

Accept a username for POP3 authentication.

Note:

This command is only allowed for sessions which have not yet been authenticated. If the username has already been supplied pre-authentication, the old value will be overwritten with the new one.

Parameters:

con the POP3 client connection issuing the command. This function returns no value.

Definition at line 131 of file pop.c.

References con_write_bl(), pop_invalid(), pop_user_parse(), and st_cleanup.

5.159.1.30 stringer_t* pop_user_parse (connection_t * con)

Parse a POP3 USER command.

Note:

The username can be at most 255 characters and will be returned in lowercase. This function will stop reading in username characters when an invalid character is encountered.

Parameters:

con the POP3 client connection that issued the USER command.

Returns:

a managed string containing the validated username, or NULL on failure.

Definition at line 257 of file parse.c.

References length, log_pedantic, lower_chr(), st_data_get(), st_import(), and st_length_get().

Referenced by pop_user().

5.160 src/network/read.c File Reference

```
#include "magma.h"
```

Functions

- `int64_t con_read_line (connection_t *con, bool_t block)`
Read a line of input from a network connection.
- `int64_t con_read (connection_t *con)`
Read data from a network connection, and store the data in the connection context buffer.
- `int64_t client_read_line (client_t *client)`
Read a line of input from a network client session.
- `int64_t client_read (client_t *client)`
Read data from a network connection, and store the data in the connection context buffer.

5.160.1 Function Documentation

5.160.1.1 `int64_t client_read (client_t * client)`

Read data from a network connection, and store the data in the connection context buffer. [read.c](#)

Returns:

-1 on general failure, -2 if the connection was reset, or the amount of data that was read.

Definition at line 304 of file `read.c`.

References `bytes`, `client_status()`, `ip_presentation()`, `log_pedantic`, `MANAGEDBUF`, `MEMORYBUF`, `mm_move()`, `pl_length_get()`, `pl_null()`, `st_avail_get()`, `st_char_get()`, `st_data_get()`, `st_length_get()`, `st_length_int()`, `st_length_set()`, `status`, `tcp_status()`, `tls_cipher()`, `tls_error()`, and `tls_read()`.

5.160.1.2 `int64_t client_read_line (client_t * client)`

Read a line of input from a network client session.

Returns:

-1 on general failure, -2 if the connection was reset, or the length of the current line of input, including the trailing new line character.

Definition at line 191 of file `read.c`.

References `bytes`, `ip_presentation()`, `line_pl_st()`, `log_pedantic`, `MANAGEDBUF`, `MEMORYBUF`, `mm_move()`, `pl_empty()`, `pl_length_get()`, `pl_null()`, `st_avail_get()`, `st_char_get()`, `st_data_get()`, `st_empty`, `st_length_get()`, `st_length_int()`, `st_length_set()`, `status`, `tcp_status()`, `tls_cipher()`, `tls_error()`, and `tls_read()`.

Referenced by `smtp_client_close()`, `smtp_client_connect()`, `smtp_client_send_data()`, `smtp_client_send_helo()`, `smtp_client_send_mailfrom()`, `smtp_client_send_nullfrom()`, and `smtp_client_send_rcptto()`.

5.160.1.3 int64_t con_read (connection_t * con)

Read data from a network connection, and store the data in the connection context buffer.

Note:

This function handles reading data from both regular and SSL connections. If the connection's network buffer hasn't been allocated, it will be initialized.

Parameters:

con a pointer to the connection object from which the data will be read.

Returns:

-1 on general failure, -2 if the connection was reset, or the amount of data that was read.

Definition at line 108 of file read.c.

References bytes, con_init_network_buffer(), con_status(), mm_move(), pl_length_get(), pl_null(), st_avail_get(), st_char_get(), st_data_get(), st_length_get(), st_length_set(), status, tcp_read(), and tls_read().

Referenced by http_body(), imap_parse_literal(), smtp_data_finish(), and smtp_data_read().

5.160.1.4 int64_t con_read_line (connection_t * con, bool_t block)

Read a line of input from a network connection.

Note:

This function handles reading data from both regular and ssl connections. This function continually attempts to read incoming data from the specified connection until a terminated line of input is received. If a new line is read, the length of that line is returned to the caller, including the trailing . If the read returns -1 and wasn't caused by a syscall interruption or blocking error, -1 is returned, and the connection status is set to -1. If the read returns 0 and wasn't caused by a syscall interruption or blocking error, -2 is returned, and the connection status is set to 2. Once a character is reached, the length of the current line of input is returned to the user, and the connection status is set to 1.

Parameters:

con the network connection across which the line of data will be read.

Returns:

-1 on general failure, -2 if the connection was reset, or the length of the current line of input, including the trailing new line character.

Definition at line 21 of file read.c.

References bytes, con_init_network_buffer(), con_status(), line_pl_st(), mm_move(), pl_empty(), pl_length_get(), pl_null(), st_avail_get(), st_char_get(), st_data_get(), st_empty, st_length_get(), st_length_set(), status, tcp_read(), and tls_read().

Referenced by dmtpl_process(), http_process(), imap_parse_literal(), imap_process(), molten_parse(), pop_process(), smtp_auth_login(), smtp_auth_plain(), and smtp_process().

5.161 src/network/reverse.c File Reference

```
#include "magma.h"
```

Functions

- void [con_reverse_enqueue](#) ([connection_t](#) *con)
Queue a reverse DNS lookup on the specified connection, if one hasn't been performed.
- void [con_reverse_domain](#) ([connection_t](#) *con, [stringer_t](#) *domain, [int_t](#) status)
Set the domain name and reverse lookup status of a connection.
- void [con_reverse_status](#) ([connection_t](#) *con, [int_t](#) status)
- [stringer_t](#) * [con_reverse_check](#) ([connection_t](#) *con, [uint32_t](#) timeout)
Poll a connection periodically to see if a reverse DNS lookup operation has completed.
- void [con_reverse_lookup](#) ([connection_t](#) *con)
Perform a reverse DNS lookup on the remote end of a connection, and save the hostname.

5.161.1 Function Documentation

5.161.1.1 [stringer_t](#)* [con_reverse_check](#) ([connection_t](#) * con, [uint32_t](#) timeout)

Poll a connection periodically to see if a reverse DNS lookup operation has completed. [reverse.c](#)

Note:

Polling will occur every 100ms, for as many seconds as specified by the user.

Parameters:

con the specified connection object to be polled that is the target of the DNS lookup.
timeout the number of seconds to wait for the lookup operation to complete.

Returns:

NULL on failure, or a pointer to a managed string containing the connection's peer hostname on success.

Definition at line 75 of file reverse.c.

References [mutex_lock\(\)](#), [mutex_unlock\(\)](#), [REVERSE_COMPLETE](#), [REVERSE_PENDING](#), and [status](#).

Referenced by [mail_add_inbound_headers\(\)](#), and [mail_add_outbound_headers\(\)](#).

5.161.1.2 void [con_reverse_domain](#) ([connection_t](#) * con, [stringer_t](#) * domain, [int_t](#) status)

Set the domain name and reverse lookup status of a connection.

Note:

Possible values for status include [REVERSE_ERROR](#), [REVERSE_EMPTY](#), [REVERSE_PENDING](#), and [REVERSE_COMPLETE](#).

Parameters:

domain the new value of the connection's hostname.

status the new value of the connection's reverse lookup status.

Returns:

This function returns no value.

Definition at line 42 of file reverse.c.

References mutex_lock(), and mutex_unlock().

Referenced by con_reverse_lookup().

5.161.1.3 void con_reverse_enqueue (connection_t * con)

Queue a reverse DNS lookup on the specified connection, if one hasn't been performed.

Parameters:

con the connection object to be examined.

Returns:

This function returns no value.

Definition at line 15 of file reverse.c.

References con_reverse_lookup(), enqueue(), mutex_lock(), mutex_unlock(), REVERSE_EMPTY, and REVERSE_PENDING.

Referenced by dmtip_init(), http_init(), imap_init(), and smtp_init().

5.161.1.4 void con_reverse_lookup (connection_t * con)

Perform a reverse DNS lookup on the remote end of a connection, and save the hostname.

Parameters:

con the connection object to be queried.

Returns:

This function returns no value.

Definition at line 107 of file reverse.c.

References con_destroy(), con_reverse_domain(), con_reverse_status(), MEMORYBUF, ns_length_get(), REVERSE_COMPLETE, REVERSE_ERROR, and st_import().

Referenced by con_reverse_enqueue().

5.161.1.5 void con_reverse_status (connection_t * con, int_t status)

Definition at line 59 of file reverse.c.

References mutex_lock(), and mutex_unlock().

Referenced by con_reverse_lookup().

5.162 src/network/sessions.h File Reference

Enumerations

- enum { [SESSION_STATE_TERMINATED](#) = -1, [SESSION_STATE_NEUTRAL](#) = 0, [SESSION_STATE_AUTHENTICATED](#) = 1 }

Functions

- struct [__attribute__](#) ((packed))

Variables

- [attachment_t](#)
- [composition_t](#)
- [session_t](#)

5.162.1 Enumeration Type Documentation

5.162.1.1 anonymous enum

Enumerator:

SESSION_STATE_TERMINATED
SESSION_STATE_NEUTRAL
SESSION_STATE_AUTHENTICATED

Definition at line 11 of file sessions.h.

5.162.2 Function Documentation

5.162.2.1 struct [__attribute__](#) ((packed)) [read]

Definition at line 29 of file sessions.h.

References [__attribute__::agent](#), [__attribute__::host](#), [inx_t](#), [lock](#), [meta_user_t](#), [number](#), [__attribute__::request](#), and [__attribute__::user](#).

5.162.3 Variable Documentation

5.162.3.1 [attachment_t](#)

Definition at line 21 of file sessions.h.

Referenced by [portal_endpoint_attachments_add\(\)](#), [portal_endpoint_attachments_remove\(\)](#), [portal_get_upload_attachment\(\)](#), [portal_smtp_create_data\(\)](#), and [portal_upload\(\)](#).

5.162.3.2 [composition_t](#)

Definition at line 27 of file sessions.h.

Referenced by [portal_endpoint_attachments_add\(\)](#), [portal_endpoint_attachments_remove\(\)](#), [portal_endpoint_messages_compose\(\)](#), [portal_endpoint_messages_send\(\)](#), and [portal_get_upload_attachment\(\)](#).

5.162.3.3 session_t

Definition at line 67 of file sessions.h.

Referenced by obj_cache_prune(), and sess_create().

5.163 src/objects/sessions/sessions.h File Reference

Functions

- `session_t * sess_create (connection_t *con, stringer_t *path, stringer_t *application)`
sessions.c
- `void sess_destroy (session_t *sess)`
Destroy a web session and its associated data.
- `int_t sess_get (connection_t *con, stringer_t *application, stringer_t *path, stringer_t *token)`
Try to retrieve the session associated with a client connection's supplied cookie.
- `uint64_t sess_key (void)`
Generate a random 12 digit 64-bit web session key.
- `uint64_t sess_number (void)`
Reserve a unique web session identifier.
- `void sess_ref_add (session_t *sess)`
Increment the web session's reference counter and update its timestamp.
- `void sess_ref_dec (session_t *sess)`
Decrement the web session's reference counter and update its timestamp.
- `time_t sess_ref_stamp (session_t *sess)`
Get the timestamp for a web session's reference counter.
- `uint64_t sess_ref_total (session_t *sess)`
Get the reference count of a web session.
- `bool_t sess_refresh_check (session_t *sess)`
Check to see if a web session is ready to be refreshed, and if so, disarm its trigger and update its refresh stamp.
- `void sess_refresh_flush (session_t *sess)`
Update a web session's refresh stamp and prevent redundant refreshing of a web session.
- `time_t sess_refresh_stamp (session_t *sess)`
Get the timestamp for the last time the session was refreshed.
- `void sess_release (session_t *sess)`
Release a web session.
- `void sess_serial_check (session_t *sess, uint64_t object)`
Queue a web session refresh if there is stale data.
- `stringer_t * sess_token (session_t *sess)`
Securely generate a unique zbase32-encoded token for a session.
- `void sess_trigger (session_t *sess)`
Set a web session's trigger so it will be refreshed as soon as possible.
- `void sess_update (session_t *sess)`

Update a session's underlying data and its refresh timestamp.

- void `sess_release_attachment` (`attachment_t` *attachment)

Free an attachment object.

- void `sess_release_composition` (`composition_t` *comp)

Free a composition object.

5.163.1 Function Documentation

5.163.1.1 `session_t`* `sess_create` (`connection_t` *con, `stringer_t` *path, `stringer_t` *application)

sessions.c sessions.c

Note:

The session stores the following data points: remote IP address, request path, application name, the specified http hostname, the remote client's user agent string, the server's host number, a unique session id, the server's current timestamp, a randomly- generated session key for authentication, and an encrypted token for the session returned to the user as a cookie.

Parameters:

con a pointer to the connection underlying the web session.

path a pointer to a managed string containing the pathname of the generating request (should be "/portal/camel").

application a pointer to a managed string containing the name of the parent application of the session (should be "portal").

Returns:

NULL on failure or a pointer to a newly allocated session object for the specified connection.

Definition at line 371 of file sessions.c.

References `con_addr()`, `con_secure()`, `CONTIGUOUS`, `HEAP`, `magma_t::host`, `inx_alloc()`, `inx_insert()`, `ip_copy()`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `magma`, `MANAGED_T`, `MEMORYBUF`, `mm_alloc()`, `mm_free()`, `magma_t::number`, `objects`, `sess_destroy()`, `sess_key()`, `sess_number()`, `sess_ref_add()`, `sess_ref_dec()`, `sess_release_composition()`, `sess_token()`, `session_t`, `object_cache_t::sessions`, `st_dup_opts()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `json_api_dispatch()`, and `portal_endpoint()`.

5.163.1.2 `void sess_destroy` (`session_t` *sess)

Destroy a web session and its associated data.

Parameters:

sess a pointer to the web session to be destroyed.

Returns:

This function returns no value.

Definition at line 59 of file sessions.c.

References `inx_cleanup()`, `meta_inx_remove()`, `META_PROTOCOL_WEB`, `mm_free()`, `mutex_destroy()`, and `st_cleanup`.

Referenced by `obj_cache_start()`, and `sess_create()`.

5.163.1.3 int_t sess_get (connection_t * con, stringer_t * application, stringer_t * path, stringer_t * token)

Try to retrieve the session associated with a client connection's supplied cookie.

Parameters:

- con* a pointer to the connection object sending the cookie.
- application* a managed string containing the application associated with the session.
- path* a managed string containing the path associated with the session.
- token* the encrypted user token retrieved from the supplied http cookie.

Returns:

1 if the cookie was found and valid, or one of the following values on failure: 0 = Session not found. -1 = Server error. -2 = Invalid token. -3 = Security violation / incorrect user-agent. -4 = Security violation / incorrect session key. -5 = Security violation / incorrect source address. -6 = Session terminated by logout. -7 = Session timed out.

Most session attributes need simple equality comparison, except for timeout checking. Make sure not to validate against a stale session that should have already timed out (which will have to be determined dynamically).

Definition at line 434 of file sessions.c.

References con_addr(), con_secure(), deprecated_scramble_decrypt(), deprecated_scramble_import(), magma_t::http, inx_delete(), inx_find(), inx_lock_read(), inx_unlock(), ip_addr_eq(), log_error, log_pedantic, M_TYPE_UINT64, magma, MEMORYBUF, number, objects, magma_t::secure, sess_ref_add(), sess_ref_dec(), sess_refresh_stamp(), sess_update(), object_cache_t::sessions, magma_t::sessions, st_char_get(), st_cleanup, st_cmp_cs_eq(), st_data_get(), st_free(), st_length_int(), multi_t::u64, multi_t::val, and zbase32_decode().

Referenced by http_parse_context().

5.163.1.4 uint64_t sess_key (void)

Generate a random 12 digit 64-bit web session key.

Returns:

the randomly generated web session key as an unsigned 64 bit integer.

Definition at line 22 of file sessions.c.

References log_pedantic, rand_get_uint64(), and uint64_digits().

Referenced by sess_create().

5.163.1.5 uint64_t sess_number (void)

Reserve a unique web session identifier.

Returns:

a number containing a unique web session identifier.

Definition at line 43 of file sessions.c.

References mutex_lock(), mutex_unlock(), and sessions.

Referenced by sess_create().

5.163.1.6 void sess_ref_add (session_t * sess)

Increment the web session's reference counter and update its timestamp.

Parameters:

sess a pointer to the web session to be updated.

Returns:

This function returns no value.

Definition at line 87 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_create(), and sess_get().

5.163.1.7 void sess_ref_dec (session_t * sess)

Decrement the web session's reference counter and update its timestamp.

Parameters:

sess a pointer to the web session to be updated.

Returns:

This function returns no value.

Definition at line 112 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_create(), sess_get(), and sess_release().

5.163.1.8 time_t sess_ref_stamp (session_t * sess)

Get the timestamp for a web session's reference counter.

Parameters:

sess a pointer to the web session to be queried.

Returns:

the last-modified UTC timestamp value of the specified web session.

Definition at line 162 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by obj_cache_prune().

5.163.1.9 uint64_t sess_ref_total (session_t * sess)

Get the reference count of a web session.

Parameters:

sess a pointer to the web session to be queried.

Returns:

the total number of references to the specified web session.

Definition at line 137 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by obj_cache_prune().

5.163.1.10 bool_t sess_refresh_check (session_t * sess)

Check to see if a web session is ready to be refreshed, and if so, disarm its trigger and update its refresh stamp.

Note:

The caller needs to make immediate use of this function's return value because the refresh trigger will be cleared if it was previously set. Any session longer than 2 minutes will be marked as ready for refresh.

Parameters:

a pointer to the web session to be queried.

Returns:

true if the web session is ready to be refreshed, or false if it is not.

Definition at line 217 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_release().

5.163.1.11 void sess_refresh_flush (session_t * sess)

Update a web session's refresh stamp and prevent redundant refreshing of a web session.

Parameters:

sess a pointer to the web session to be touched.

Returns:

This function returns no value.

Definition at line 180 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_update().

5.163.1.12 time_t sess_refresh_stamp (session_t * sess)

Get the timestamp for the last time the session was refreshed.

Parameters:

sess a pointer to the web session to be queried.

Returns:

the last-refreshed UTC timestamp value of the specified web session.

Definition at line 197 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_get().

5.163.1.13 void sess_release (session_t * sess)

Release a web session.

Note:

If the web session should have been refreshed, it will be refreshed before the reference counter is decremented.

Parameters:

sess a pointer to the web session to be released.

Returns:

This function returns no value.

Definition at line 302 of file sessions.c.

References requeue(), sess_ref_dec(), sess_refresh_check(), and sess_update().

Referenced by http_session_reset().

5.163.1.14 void sess_release_attachment (attachment_t * attachment)

Free an attachment object.

Note:

This is an inx helper function.

Parameters:

attachment a pointer to the attachment object to be destroyed.

Returns:

This function returns no value.

Definition at line 321 of file sessions.c.

References mm_free(), and st_cleanup.

Referenced by portal_endpoint_attachments_add(), and portal_endpoint_messages_compose().

5.163.1.15 void sess_release_composition (composition_t * comp)

Free a composition object.

Note:

This is an inx helper function.

Parameters:

comp a pointer to the composition object to be destroyed.

Returns:

This function returns no value.

Definition at line 337 of file sessions.c.

References inx_cleanup(), and mm_free().

Referenced by portal_endpoint_messages_compose(), and sess_create().

5.163.1.16 void sess_serial_check (session_t * sess, uint64_t object)

Queue a web session refresh if there is stale data.

Parameters:

sess a pointer to the web session to be queried.

object the value of the object in the cache to be checked (can include OBJECT_CONTACTS and OBJECT_FOLDERS).

Definition at line 352 of file sessions.c.

References meta_user_serial_check(), and sess_trigger().

Referenced by portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_edit(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), portal_folder_contacts_add(), portal_folder_contacts_remove(), portal_folder_mail_add(), and portal_folder_mail_remove().

5.163.1.17 stringer_t* sess_token (session_t * sess)

Securely generate a unique zbase32-encoded token for a session.

Parameters:

sess a pointer to the input web session.

Returns:

a managed string containing the generated web session token.

Definition at line 241 of file sessions.c.

References deprecated_scramble_encrypt(), deprecated_scramble_free(), deprecated_scramble_total_length(), log_pedantic, magma, PLACER, magma_t::secure, magma_t::sessions, st_cleanup, st_merge, and zbase32_encode().

Referenced by sess_create().

5.163.1.18 void sess_trigger (session_t * sess)

Set a web session's trigger so it will be refreshed as soon as possible.

Parameters:

sess a pointer to the web session to be refreshed.

Returns:

This function returns no value.

Definition at line 285 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by portal_endpoint_folders_rename(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), and sess_serial_check().

5.163.1.19 void sess_update (session_t * sess)

Update a session's underlying data and its refresh timestamp.

Note:

This function ensures the session's associated meta user data, mail folders and messages, and contact folders and contact entries are all current.

Parameters:

sess a pointer to the session object to be updated.

Returns:

This function returns no value.

Definition at line 264 of file sessions.c.

References `meta_messages_update()`, `META_NEED_LOCK`, `meta_update_aliases()`, `meta_update_contacts()`, `meta_update_folders()`, `meta_update_user()`, and `sess_refresh_flush()`.

Referenced by `sess_get()`, and `sess_release()`.

5.164 src/network/smtp.h File Reference

Data Structures

- struct [smtp_message_t](#)
- struct [smtp_recipients_t](#)
- struct [smtp_inbound_filter_t](#)
- struct [smtp_inbound_prefs_t](#)
- struct [smtp_outbound_prefs_t](#)
- struct [smtp_session_t](#)

Enumerations

- enum {
[SMTP_ACTION_ERROR](#) = -1, [SMTP_ACTION_UNDEFINED](#) = 0, [SMTP_ACTION_DELETE](#) = 1, [SMTP_ACTION_REJECT](#) = 2,
[SMTP_ACTION_BOUNCE](#) = 3, [SMTP_ACTION_MARK](#) = 4, [SMTP_ACTION_MARK_READ](#) = 5, [SMTP_MARK_NONE](#) = 0,
[SMTP_MARK_READ](#) = 1, [SMTP_MARK_PHISH](#) = 2, [SMTP_MARK_VIRUS](#) = 4, [SMTP_MARK_SPAM](#) = 8,
[SMTP_MARK_RBL](#) = 16, [SMTP_MARK_SPOOF](#) = 32, [SMTP_MARK_FILTERED](#) = 64, [SMTP_FILTER_TYPE_EXACT](#) = 0,
[SMTP_FILTER_TYPE_CONTAINS](#) = 1, [SMTP_FILTER_TYPE_STARTS](#) = 2, [SMTP_FILTER_TYPE_ENDS](#) = 4, [SMTP_FILTER_TYPE_REGEX](#) = 8,
[SMTP_FILTER_LOCATION_HEADER](#) = 1, [SMTP_FILTER_LOCATION_BODY](#) = 2, [SMTP_FILTER_LOCATION_FIELD](#) = 4,
[SMTP_FILTER_LOCATION_ENTIRE](#) = 8,
[SMTP_FILTER_ACTION_NONE](#) = 0, [SMTP_FILTER_ACTION_MOVE](#) = 1, [SMTP_FILTER_ACTION_LABEL](#) = 2, [SMTP_FILTER_ACTION_DELETE](#) = 4,
[SMTP_FILTER_ACTION_MARK_READ](#) = 8, [SMTP_OUTCOME_SUCESS](#) = 0, [SMTP_OUTCOME_PERM_FAILURE](#) = 1,
[SMTP_OUTCOME_TEMP_SERVER](#) = 2,
[SMTP_OUTCOME_TEMP_OVERQUOTA](#) = 4, [SMTP_OUTCOME_TEMP_LOCKED](#) = 8, [SMTP_OUTCOME_BOUNCE_SPF](#) = 16,
[SMTP_OUTCOME_BOUNCE_DKIM](#) = 32,
[SMTP_OUTCOME_BOUNCE_VIRUS](#) = 64, [SMTP_OUTCOME_BOUNCE_PHISH](#) = 128, [SMTP_OUTCOME_BOUNCE_SPAM](#) = 256,
[SMTP_OUTCOME_BOUNCE_RBL](#) = 512 }

5.164.1 Enumeration Type Documentation

5.164.1.1 anonymous enum

Enumerator:

[*SMTP_ACTION_ERROR*](#)
[*SMTP_ACTION_UNDEFINED*](#)
[*SMTP_ACTION_DELETE*](#)
[*SMTP_ACTION_REJECT*](#)
[*SMTP_ACTION_BOUNCE*](#)
[*SMTP_ACTION_MARK*](#)
[*SMTP_ACTION_MARK_READ*](#)
[*SMTP_MARK_NONE*](#)
[*SMTP_MARK_READ*](#)
[*SMTP_MARK_PHISH*](#)
[*SMTP_MARK_VIRUS*](#)
[*SMTP_MARK_SPAM*](#)

SMTP_MARK_RBL
SMTP_MARK_SPOOF
SMTP_MARK_FILTERED
SMTP_FILTER_TYPE_EXACT
SMTP_FILTER_TYPE_CONTAINS
SMTP_FILTER_TYPE_STARTS
SMTP_FILTER_TYPE_ENDS
SMTP_FILTER_TYPE_REGEX
SMTP_FILTER_LOCATION_HEADER
SMTP_FILTER_LOCATION_BODY
SMTP_FILTER_LOCATION_FIELD
SMTP_FILTER_LOCATION_ENTIRE
SMTP_FILTER_ACTION_NONE
SMTP_FILTER_ACTION_MOVE
SMTP_FILTER_ACTION_LABEL
SMTP_FILTER_ACTION_DELETE
SMTP_FILTER_ACTION_MARK_READ
SMTP_OUTCOME_SUCCESS
SMTP_OUTCOME_PERM_FAILURE
SMTP_OUTCOME_TEMP_SERVER
SMTP_OUTCOME_TEMP_OVERQUOTA
SMTP_OUTCOME_TEMP_LOCKED
SMTP_OUTCOME_BOUNCE_SPF
SMTP_OUTCOME_BOUNCE_DKIM
SMTP_OUTCOME_BOUNCE_VIRUS
SMTP_OUTCOME_BOUNCE_PHISH
SMTP_OUTCOME_BOUNCE_SPAM
SMTP_OUTCOME_BOUNCE_RBL

Definition at line 11 of file smtp.h.

5.165 src/servers/smtp/smtp.h File Reference

Functions

- [int_t smtp_accept_message](#) ([connection_t](#) *con, [smtp_inbound_prefs_t](#) *prefs)
accept.c
- [int_t smtp_rollout](#) ([smtp_inbound_prefs_t](#) *prefs)
Delete the oldest mail message owned by a user until their storage usage falls below their storage quota.
- [int_t smtp_store_message](#) ([smtp_inbound_prefs_t](#) *prefs, [stringer_t](#) **local)
Store a received SMTP message as a generic mail message, both on disk and in the database.
- [bool_t smtp_store_spamsig](#) ([smtp_inbound_prefs_t](#) *prefs, [int_t](#) spam)
Generate a random key for a spam signature and store it in the database.
- [int_t smtp_check_filters](#) ([smtp_inbound_prefs_t](#) *prefs, [stringer_t](#) **local)
checkers.c
- [int_t smtp_check_greylist](#) ([connection_t](#) *con, [smtp_inbound_prefs_t](#) *prefs)
Check to see if a transmitting address is in a user's greylist.
- [int_t smtp_check_rbl](#) ([connection_t](#) *con)
Check the SMTP connection's remote address against a collection of real-time blacklists.
- [bool_t smtp_add_bypass_entry](#) ([stringer_t](#) *subnet)
Add an entry to the SMTP subnet bypass list.
- [bool_t smtp_bypass_check](#) ([connection_t](#) *con)
Check if a connection should bypass certain SMTP checks.
- [void smtp_requeue](#) ([connection_t](#) *con)
commands.c
- [int_t smtp_compare](#) (const void *compare, const void *command)
- [void smtp_process](#) ([connection_t](#) *con)
The main entry point in the SMTP server for processing commands issued by clients.
- [void smtp_sort](#) (void)
Sort the SMTP command table to be ready for binary searches.
- [int_t smtp_check_authorized_from](#) ([uint64_t](#) usernum, [stringer_t](#) *address)
datatier.c
- [int_t smtp_check_receive_quota](#) ([connection_t](#) *con, [smtp_inbound_prefs_t](#) *prefs)
Check the user's received statistics from the database to see if receiving a message would result in a quota overage.
- [int_t smtp_check_transmit_quota](#) ([uint64_t](#) usernum, [size_t](#) num_recipients, [smtp_outbound_prefs_t](#) *prefs)
Check to see if a user's current mail send request would push them over their daily transmission quota.
- [int_t smtp_fetch_authorization](#) ([stringer_t](#) *username, [stringer_t](#) *verification, [smtp_outbound_prefs_t](#) **output)
Check if a user is authorized to send messages, and retrieve the user's outbound smtp preferences.

- [stringer_t * smtp_fetch_autoreply](#) (uint64_t autoreply, uint64_t usernum)
Fetch a specified auto-reply message for a user.
- [int_t smtp_fetch_inbound](#) (stringer_t *address, smtp_inbound_prefs_t **output)
Fetch a user's SMTP preferences for inbound mail.
- [table_t * smtp_fetch_rollmessages](#) (uint64_t usernum)
Retrieve a list, at a maximum of 20 entries, of the oldest messages owned by a user.
- [int_t smtp_get_action](#) (chr_t *string, size_t length)
Parse an smtp event action string from the Dispatch table.
- [uint64_t smtp_insert_spamsig](#) (smtp_inbound_prefs_t *prefs, uint64_t key, [int_t](#) code)
Store a spam signature in the database.
- [void smtp_update_receive_stats](#) (connection_t *con, smtp_inbound_prefs_t *prefs)
Update the receiving statistics and per-user log tables in the database for a successfully received smtp message.
- [void smtp_update_transmission_stats](#) (connection_t *con)
Update the transmission and per-user log tables in the database for a successfully sent smtp message.
- [void smtp_auth_login](#) (connection_t *con)
smtp.c
- [void smtp_auth_plain](#) (connection_t *con)
- [void smtp_data](#) (connection_t *con)
- [void smtp_disabled](#) (connection_t *con)
A stub function for an SMTP command that has not been implemented.
- [void smtp_ehlo](#) (connection_t *con)
Process an SMTP EHLO command.
- [void smtp_helo](#) (connection_t *con)
Process an SMTP HELO command.
- [void smtp_init](#) (connection_t *con)
The start of the protocol handler for the SMTP server.
- [void smtp_invalid](#) (connection_t *con)
A function that is executed when an invalid SMTP command is executed.
- [void smtp_mail_from](#) (connection_t *con)
Specify the identity of a message's sender, in response to an SMTP MAIL FROM command.
- [void smtp_noop](#) (connection_t *con)
Perform an SMTP NOOP (no-operation) command.
- [void smtp_quit](#) (connection_t *con)
Gracefully terminate an SMTP session, especially in response to an SMTP QUIT command.
- [void smtp_rcpt_to](#) (connection_t *con)
- [void smtp_rset](#) (connection_t *con)
Reset the SMTP session, in response to an SMTP RSET command.

- void [smtp_starttls](#) ([connection_t](#) *con)
TODO: Review error messages and update them with the appropriate response code.
- void [submission_init](#) ([connection_t](#) *con)
- [stringer_t](#) * [smtp_parse_auth](#) ([stringer_t](#) *data)
parse.c
- [stringer_t](#) * [smtp_parse_helo_domain](#) ([connection_t](#) *con)
Parse the domain specified as the parameter to an SMTP HELO or EHLO command.
- [stringer_t](#) * [smtp_parse_mail_from_path](#) ([connection_t](#) *con)
- [stringer_t](#) * [smtp_parse_rcpt_to](#) ([connection_t](#) *con)
- void [smtp_client_close](#) ([client_t](#) *client)
relay.c
- [client_t](#) * [smtp_client_connect](#) ([int_t](#) premium)
Connect to a randomly selected mail relay server, and wait for a successful banner message.
- [int_t](#) [smtp_client_send_data](#) ([client_t](#) *client, [stringer_t](#) *message, [bool_t](#) dotstuffed)
Issue a DATA command to an smtp server, and wait for a successful response.
- [int_t](#) [smtp_client_send_helo](#) ([client_t](#) *client)
Issue a EHLO command to an smtp server, or fall back to HELO, and wait for a successful response.
- [int_t](#) [smtp_client_send_mailfrom](#) ([client_t](#) *client, [stringer_t](#) *mailfrom, [size_t](#) send_size)
Issue a MAIL FROM command to an smtp server, and wait for a successful response.
- [int_t](#) [smtp_client_send_nullfrom](#) ([client_t](#) *client)
Issue a MAIL FROM command to an smtp server with a null sender, and wait for a successful response.
- [int_t](#) [smtp_client_send_rcptto](#) ([client_t](#) *client, [stringer_t](#) *rcptto)
Issue a RCPT TO command to an smtp server, and wait for a successful response.
- void [smtp_add_inbound](#) ([connection_t](#) *con, [smtp_inbound_prefs_t](#) *inbound)
session.c
- void [smtp_add_outbound](#) ([connection_t](#) *con, [smtp_outbound_prefs_t](#) *outbound)
Attach a set of SMTP outbound mail preferences to an SMTP client connection.
- [bool_t](#) [smtp_add_recipient](#) ([connection_t](#) *con, [stringer_t](#) *address)
Add an entry to an SMTP session's recipients list for outbound/relayed mail.
- [bool_t](#) [smtp_check_duplicate_recipient](#) ([connection_t](#) *con, [uint64_t](#) usernum)
Check whether the usernum on the inbound structure has already been used.
- void [smtp_free_inbound](#) ([smtp_inbound_prefs_t](#) *inbound)
Free a list of SMTP inbound mail preferences and its underlying data.
- void [smtp_free_outbound](#) ([smtp_outbound_prefs_t](#) *outbound)
Free a set of SMTP outbound mail preferences and its underlying data.
- void [smtp_free_recipients](#) ([smtp_recipients_t](#) *recipients)

Free a list of SMTP recipients and its underlying data.

- void `smtp_list_free_filter` (`smtp_inbound_filter_t` *filter)

Free an SMTP inbound filter and its underlying data.

- void `smtp_session_destroy` (`connection_t` *con)

Destroy the data associated with an SMTP session.

- void `smtp_session_reset` (`connection_t` *con)

Reset an SMTP session to its initialized state.

- `int_t` `smtp_bounce` (`connection_t` *con)

transmit.c

- `int_t` `smtp_forward_message` (`server_t` *server, `stringer_t` *sender, `stringer_t` *address, `stringer_t` *message, `stringer_t` *id, `int_t` mark, `uint64_t` signum, `uint64_t` sigkey)

- `int_t` `smtp_relay_message` (`connection_t` *con, `stringer_t` **result)

Relay an outbound smtp message for a user.

- `int_t` `smtp_reply` (`stringer_t` *from, `stringer_t` *to, `uint64_t` usernum, `uint64_t` autoreply, `int_t` spf, `int_t` dkim)

- `int_t` `smtp_send_message` (`stringer_t` *to, `stringer_t` *from, `stringer_t` *message)

Relay an outbound smtp message for the user.

5.165.1 Function Documentation

5.165.1.1 `int_t smtp_accept_message` (`connection_t` *con, `smtp_inbound_prefs_t` *prefs)

`accept.c`

Definition at line 185 of file `accept.c`.

References `magma_t::abuse`, `magma_t::admin`, `smtp_inbound_prefs_t::autoreply`, `smtp_inbound_prefs_t::bounces`, `magma_t::contact`, `smtp_inbound_prefs_t::dkim`, `dkim_signature_verify()`, `smtp_inbound_prefs_t::dkimaction`, `dspam_check()`, `smtp_inbound_prefs_t::filters`, `smtp_inbound_prefs_t::foldernum`, `smtp_inbound_prefs_t::forwarded`, `smtp_inbound_prefs_t::inbox`, `log_error`, `log_pedantic`, `magma`, `mail_add_inbound_headers()`, `smtp_inbound_prefs_t::mark`, `smtp_inbound_prefs_t::overquota`, `smtp_inbound_prefs_t::phish`, `smtp_inbound_prefs_t::phishaction`, `PLACER`, `smtp_inbound_prefs_t::rbl`, `smtp_inbound_prefs_t::rblaction`, `smtp_inbound_prefs_t::rcptto`, `smtp_inbound_prefs_t::rcv_size_limit`, `smtp_inbound_prefs_t::rollout`, `smtp_inbound_prefs_t::signum`, `SMTP_ACTION_BOUNCE`, `SMTP_ACTION_DELETE`, `SMTP_ACTION_MARK`, `SMTP_ACTION_MARK_READ`, `smtp_check_filters()`, `smtp_forward_message()`, `SMTP_MARK_NONE`, `SMTP_MARK_PHISH`, `SMTP_MARK_RBL`, `SMTP_MARK_READ`, `SMTP_MARK_SPAM`, `SMTP_MARK_SPOOF`, `SMTP_MARK_VIRUS`, `SMTP_OUTCOME_BOUNCE_DKIM`, `SMTP_OUTCOME_BOUNCE_PHISH`, `SMTP_OUTCOME_BOUNCE_RBL`, `SMTP_OUTCOME_BOUNCE_SPAM`, `SMTP_OUTCOME_BOUNCE_SPF`, `SMTP_OUTCOME_BOUNCE_VIRUS`, `SMTP_OUTCOME_PERM_FAILURE`, `SMTP_OUTCOME_SUCESS`, `SMTP_OUTCOME_TEMP_LOCKED`, `SMTP_OUTCOME_TEMP_OVERQUOTA`, `SMTP_OUTCOME_TEMP_SERVER`, `smtp_reply()`, `smtp_rollout()`, `smtp_store_message()`, `smtp_store_spamsig()`, `smtp_update_receive_stats()`, `smtp_inbound_prefs_t::spam`, `smtp_inbound_prefs_t::spam_checked`, `smtp_inbound_prefs_t::spamaction`, `smtp_inbound_prefs_t::spamkey`, `smtp_inbound_prefs_t::spamsig`, `smtp_inbound_prefs_t::spf`, `smtp_inbound_prefs_t::spfaction`, `st_cmp_ci_eq()`, `st_cmp_cs_eq()`, `st_free()`, `st_length_get()`, `smtp_inbound_prefs_t::usernum`, `smtp_inbound_prefs_t::virus`, `virus_check()`, and `smtp_inbound_prefs_t::virusaction`.

Referenced by `smtp_data_inbound()`.

5.165.1.2 `bool_t smtp_add_bypass_entry` (`stringer_t` *subnet)

Add an entry to the SMTP subnet bypass list.

Parameters:

subnet a pointer to a managed string containing the IP address or subnet address to be bypassed for checks.

Returns:

true if the entry was valid and was added, or false on failure.

Definition at line 249 of file checkers.c.

References magma_t::bypass_subnets, inx_alloc(), inx_insert(), ip_subnet_st(), log_error, log_pedantic, M_INX_LINKED, M_TYPE_UINT64, magma, mm_alloc(), mm_free(), magma_t::smtp, st_char_get(), multi_t::type, multi_t::u64, and multi_t::val.

Referenced by config_value_set().

5.165.1.3 void smtp_add_inbound (connection_t * con, smtp_inbound_prefs_t * inbound)

session.c session.c

Parameters:

con the SMTP client connection specifying the added local recipient.

inbound a pointer to the SMTP inbound preferences object to be added to the SMTP session's inbound preferences list.

Returns:

true if the requested inbound preferences object was added successfully to the inbound preferences list, or false on failure.

Definition at line 150 of file session.c.

References smtp_inbound_prefs_t::next.

Referenced by smtp_rcpt_to().

5.165.1.4 void smtp_add_outbound (connection_t * con, smtp_outbound_prefs_t * outbound)

Attach a set of SMTP outbound mail preferences to an SMTP client connection.

Parameters:

con the SMTP client connection to which the outbound mail preferences should be attached.

outbound a pointer to the SMTP outbound mail preferences to be set for the specified connection.

Returns:

This function returns no value.

Definition at line 179 of file session.c.

Referenced by smtp_auth_login(), and smtp_auth_plain().

5.165.1.5 bool_t smtp_add_recipient (connection_t * con, stringer_t * address)

Add an entry to an SMTP session's recipients list for outbound/relayed mail.

Note:

If the recipients list does not exist, it will be created automatically.

Parameters:

- con* the SMTP client connection specifying the added recipient.
- address* a managed string containing the recipient's email address.

Returns:

true if the requested recipient was added successfully to the recipients list, or false on failure.

Definition at line 107 of file session.c.

References smtp_recipients_t::address, log_pedantic, mm_alloc(), mm_free(), smtp_recipients_t::next, and st_dupe().

Referenced by smtp_rcpt_to().

5.165.1.6 void smtp_auth_login (connection_t * con)

[smtp.c](#)

Definition at line 393 of file smtp.c.

References auth_free(), AUTH_LOCK_ABUSE, AUTH_LOCK_ADMIN, AUTH_LOCK_EXPIRED, AUTH_LOCK_USER, auth_login(), base64_decode(), cache_decrement(), cache_increment(), con_addr_presentation(), con_addr_subnet(), con_read_line(), con_write_bl(), lock_get(), lock_release(), auth_t::locked, log_info, MANAGEDBUF, pl_char_get(), pl_length_get(), PLACER, smtp_outbound_prefs_t::send_size_limit, smtp_add_outbound(), smtp_fetch_authorization(), smtp_parse_auth(), smtp_session_reset(), st_char_get(), st_cleanup, st_cmp_ci_eq(), st_empty, st_free(), st_length_int(), st_populated, st_quick(), auth_t::status, time_datestamp(), auth_t::tokens, auth_t::username, and auth_t::verification.

5.165.1.7 void smtp_auth_plain (connection_t * con)

Definition at line 224 of file smtp.c.

References auth_free(), AUTH_LOCK_ABUSE, AUTH_LOCK_ADMIN, AUTH_LOCK_EXPIRED, AUTH_LOCK_USER, auth_login(), base64_decode(), cache_decrement(), cache_increment(), con_addr_presentation(), con_addr_subnet(), con_read_line(), con_write_bl(), FOREIGNDATA, JOINTED, lock_get(), lock_release(), auth_t::locked, log_info, MANAGEDBUF, mm_copy(), pl_char_get(), pl_length_get(), PLACER, PLACER_T, placer_t, smtp_outbound_prefs_t::send_size_limit, smtp_add_outbound(), smtp_fetch_authorization(), smtp_parse_auth(), smtp_session_reset(), st_char_get(), st_cleanup, st_cmp_ci_eq(), st_empty, st_free(), st_length_get(), st_length_int(), st_populated, st_quick(), STACK, auth_t::status, time_datestamp(), tok_get_st(), auth_t::tokens, auth_t::username, and auth_t::verification.

5.165.1.8 int_t smtp_bounce (connection_t * con)

[transmit.c](#)

Definition at line 179 of file transmit.c.

References client_t, crc64_checksum(), dkim_signature_create(), magma_t::domain, log_pedantic, magma, MANAGEDBUF, smtp_inbound_prefs_t::next, number, smtp_inbound_prefs_t::outcome, rand_choices(), smtp_inbound_prefs_t::rcptto, smtp_client_close(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_nullfrom(), smtp_client_send_rcptto(), SMTP_OUTCOME_BOUNCE_DKIM, SMTP_OUTCOME_BOUNCE_PHISH, SMTP_OUTCOME_BOUNCE_RBL, SMTP_OUTCOME_BOUNCE_SPAM, SMTP_OUTCOME_BOUNCE_SPF, SMTP_OUTCOME_BOUNCE_VIRUS, SMTP_OUTCOME_PERM_FAILURE, SMTP_OUTCOME_SUCESS, SMTP_OUTCOME_TEMP_LOCKED, SMTP_OUTCOME_TEMP_OVERQUOTA, SMTP_OUTCOME_TEMP_SERVER, st_char_get(), st_cleanup, st_cmp_cs_eq(), st_free(), st_length_int(), st_merge, st_sprint(), and magma_t::system.

Referenced by smtp_data_inbound().

5.165.1.9 bool_t smtp_bypass_check (connection_t * con)

Check if a connection should bypass certain SMTP checks.

Note:

This check is run against host and/or subnet masks configured in the magma.smtp.bypass_addr option.

Parameters:

con a pointer to the connection object to be checked.

Returns:

true if the specified connection meets the SMTP bypass check or false on failure or if it does not.

Definition at line 289 of file checkers.c.

References magma_t::bypass_subnets, con_addr(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), ip_matches_subnet(), magma, and magma_t::smtp.

Referenced by smtp_init().

5.165.1.10 int_t smtp_check_authorized_from (uint64_t usernum, stringer_t * address)

datatier.c datatier.c

Note:

The authorization attempt first checks against the specified email address, and if unsuccessful, upon the domain component of the email address if wildcards are enabled for that domain.

Parameters:

usernum the numerical id of the user attempting to send the mail message.

address a pointer to a managed string containing the From address value of the mail message to be sent.

Returns:

1 if the user is authorized to send email from the specified address, or 0 otherwise.

Definition at line 761 of file datatier.c.

References domain_wildcard(), mail_domain_get(), mm_wipe(), placer_t, res_row_count(), res_table_free(), st_char_get(), st_length_get(), and stmt_get_result().

Referenced by portal_outbound_checks(), and smtp_data_outbound().

5.165.1.11 bool_t smtp_check_duplicate_recipient (connection_t * con, uint64_t usernum)

Check whether the usernum on the inbound structure has already been used.

Definition at line 80 of file session.c.

References smtp_inbound_prefs_t::next, and smtp_inbound_prefs_t::usernum.

Referenced by smtp_rcpt_to().

5.165.1.12 int_t smtp_check_filters (smtp_inbound_prefs_t * prefs, stringer_t ** local)

[checkers.c](#) Apply any user specific filters. Return -1 for errors, and -2 to delete a message. Return 1 if no action was taken, and 2 if the message was moved to a different folder, 3 if the message content was modified, and 4 if the message was marked read.

Definition at line 126 of file checkers.c.

References `data`, `smtp_inbound_prefs_t::filters`, `smtp_inbound_prefs_t::foldernum`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `length`, `log_pedantic`, `mail_header_end()`, `mail_header_fetch_all()`, `mail_mod_subject()`, `smtp_inbound_prefs_t::mark`, `MEMORYBUF`, `mm_wipe()`, `pl_data_get()`, `pl_init()`, `pl_length_get()`, `pl_null()`, `PLACER`, `placer_t`, `SMTP_FILTER_ACTION_DELETE`, `SMTP_FILTER_ACTION_LABEL`, `SMTP_FILTER_ACTION_MARK_READ`, `SMTP_FILTER_ACTION_MOVE`, `SMTP_FILTER_LOCATION_BODY`, `SMTP_FILTER_LOCATION_ENTIRE`, `SMTP_FILTER_LOCATION_FIELD`, `SMTP_FILTER_LOCATION_HEADER`, `SMTP_MARK_NONE`, `SMTP_MARK_PHISH`, `SMTP_MARK_RBL`, `SMTP_MARK_READ`, `SMTP_MARK_SPAM`, `SMTP_MARK_SPOOF`, `SMTP_MARK_VIRUS`, `st_char_get()`, `st_cleanup`, `st_cmp_ci_eq()`, `st_length_get()`, `st_length_int()`, and `smtp_inbound_prefs_t::usernum`.

Referenced by `smtp_accept_message()`.

5.165.1.13 `int_t smtp_check_greylist (connection_t * con, smtp_inbound_prefs_t * prefs)`

Check to see if a transmitting address is in a user's greylist.

Note:

The greylist is configured in the Dispatch table and specifies the minimum time, in minutes, that a transmitting smtp relay server must wait in order to be able to send more messages to the same recipient address again.

Parameters:

con the connection to have its remote address checked against the user's greylist.

prefs the smtp inbound preferences of the user

Returns:

-1 on an internal server error, 0 if the sender must wait longer, and 1 if the check was passed.

Definition at line 18 of file `checkers.c`.

References `BLOCK_T`, `cache_get()`, `cache_set()`, `con_addr_reversed()`, `CONTIGUOUS`, `smtp_inbound_prefs_t::greytime`, `HEAP`, `imap_fetch_response_t::key`, `log_pedantic`, `MANAGEDBUF`, `st_alloc_opts()`, `st_char_get()`, `st_cleanup`, `st_data_get()`, `st_length_get()`, `st_length_int()`, `st_sprint()`, `smtp_inbound_prefs_t::usernum`, and `imap_fetch_response_t::value`.

Referenced by `smtp_rcpt_to()`.

5.165.1.14 `int_t smtp_check_rbl (connection_t * con)`

Check the SMTP connection's remote address against a collection of real-time blacklists.

Note:

The connection's IP address will be checked against each of the servers configured in `magma.smtp.blacklists.domain`.

Parameters:

con the connection to have its address examined against the RBLs.

Returns:

-1 on general error, -2 if the address was blacklisted, or 1 if it passed the check.

Definition at line 77 of file `checkers.c`.

References `magma_t::blacklists`, `con_addr_reversed()`, `log_pedantic`, `magma`, `MANAGEDBUF`, `mm_wipe()`, `magma_t::smtp`, `st_char_get()`, and `st_length_int()`.

Referenced by `smtp_rcpt_to()`.

5.165.1.15 int_t smtp_check_receive_quota (connection_t * con, smtp_inbound_prefs_t * prefs)

Check the user's received statistics from the database to see if receiving a message would result in a quota overage.

Note:

The sum total of all emails received by the user over the past 24 hours is calculated from the database, and these checks are made: 1. The amount of mail messages received in the past 24 hours by the user does not exceed their daily mail received quota. 2. The messages received in the past 24 hours by the user from this subnet does not exceed the user's daily per-subnet received quota.

Parameters:

con a pointer to the connection object over which the smtp message was received.

prefs a pointer to the user's smtp inbound mail preferences.

Returns:

0 if message receipt is permitted, 1 if the general daily receiving limit was exceeded, 2 if the sending subnet's transmission limit was exceeded, or -1 if there was a general error.

Definition at line 696 of file datatier.c.

References con_addr_subnet(), smtp_inbound_prefs_t::daily_rcv_limit, smtp_inbound_prefs_t::daily_rcv_limit_ip, log_pedantic, mm_wipe(), number, res_field_block(), res_field_length(), res_field_uint64(), res_row_next(), res_table_free(), st_char_get(), st_free(), st_length_get(), stmt_get_result(), uint64_conv_bl(), and smtp_inbound_prefs_t::usernum.

Referenced by smtp_rcpt_to().

5.165.1.16 int_t smtp_check_transmit_quota (uint64_t usernum, size_t num_recipients, smtp_outbound_prefs_t * prefs)

Check to see if a user's current mail send request would push them over their daily transmission quota.

Note:

This check is performed by querying the database to see how many messages a user has sent in the past 24 hour period, and by adding the current number of recipients of the pending email request to that number to see if their quota would be exceeded.

Parameters:

con a pointer to the connection object of the user attempting to send mail.

Returns:

-1 on error, 0 if the send operation is permitted, or 1 if the send operation would result in a daily send quota overage.

Definition at line 543 of file datatier.c.

References smtp_outbound_prefs_t::daily_send_limit, log_pedantic, mm_wipe(), res_field_uint64(), res_row_next(), res_table_free(), smtp_outbound_prefs_t::sent_today, and stmt_get_result().

Referenced by portal_outbound_checks(), and smtp_data_outbound().

5.165.1.17 void smtp_client_close (client_t * client)

relay.c relay.c

Parameters:

client a pointer to the smtp client session to be closed.

Returns:

This function returns no value.

Definition at line 15 of file relay.c.

References client_close(), client_read_line(), client_write(), and PLACER.

Referenced by portal_smtp_relay_message(), smtp_bounce(), smtp_forward_message(), smtp_relay_message(), smtp_reply(), and smtp_send_message().

5.165.1.18 client_t* smtp_client_connect (int_t premium)

Connect to a randomly selected mail relay server, and wait for a successful banner message.

Parameters:

premium if set, a premium relay will be selected instead of a standard one.

Returns:

NULL on failure or a pointer to the newly established network client object connected to a mail relay on success.

Definition at line 31 of file relay.c.

References client_close(), client_connect(), client_read_line(), client_secure(), client_t, magma_t::count, magma_t::host, log_pedantic, magma, MAGMA_RELAY_INSTANCES, relay_t::name, net_set_timeout(), relay_t::port, magma_t::premium, rand_get_uint32(), magma_t::relay, relay_t::secure, st_char_get(), st_length_get(), st_length_int(), and magma_t::timeout.

Referenced by portal_smtp_relay_message(), smtp_bounce(), smtp_forward_message(), smtp_relay_message(), smtp_reply(), and smtp_send_message().

5.165.1.19 int_t smtp_client_send_data (client_t * client, stringer_t * message, bool_t dotstuffed)

Issue a DATA command to an smtp server, and wait for a successful response.

Parameters:

client a pointer to the network client to issue the DATA command.

message a pointer to a managed string containing the body of the message to be sent.

dotstuffed a boolean to where true indicates the supplied message has already been dotstuffed.

Returns:

-3 for internal errors, -2 if the remote server rejected the command, -1 on general network failure, or 1 on success.

Definition at line 243 of file relay.c.

References client_read_line(), client_write(), HEAP, JOINTED, log_pedantic, MANAGEDBUF, MAPPED_T, pl_char_get(), pl_length_int(), pl_starts_with_char(), PLACER, st_char_get(), st_cleanup, st_cmp_cs_ends(), st_dupe_opts(), st_empty, st_length_get(), st_length_int(), st_quick(), and st_replace().

Referenced by portal_smtp_relay_message(), smtp_bounce(), smtp_forward_message(), smtp_relay_message(), smtp_reply(), and smtp_send_message().

5.165.1.20 int_t smtp_client_send_helo (client_t * client)

Issue a EHLO command to an smtp server, or fall back to HELO, and wait for a successful response.

Parameters:

client a pointer to the network client to issue the remote command.

Returns:

-1 on failure or 1 on success.

Definition at line 110 of file relay.c.

References `client_print()`, `client_read_line()`, `magma_t::domain`, `magma_t::host`, `log_pedantic`, `magma`, `magma_t::name`, `ns_length_get()`, `pl_init()`, `pl_null()`, `placer_t`, `st_char_get()`, `st_empty`, `st_length_get()`, and `magma_t::system`.

Referenced by `portal_smtp_relay_message()`, `smtp_bounce()`, `smtp_forward_message()`, `smtp_relay_message()`, `smtp_reply()`, and `smtp_send_message()`.

5.165.1.21 int_t smtp_client_send_mailfrom (client_t * client, stringer_t * mailfrom, size_t send_size)

Issue a MAIL FROM command to an smtp server, and wait for a successful response.

Parameters:

client a pointer to the network client to issue the MAIL FROM command.

mailfrom a pointer to a managed string containing the address parameter for the MAIL FROM command.

send_size if greater than 0, specify the optional SIZE parameter to the MAIL FROM command.

Returns:

-2 if the remote server rejected the command, -1 on general network failure, or 1 on success.

LOW: Technically we should only be sending the size parameter if the EHLO response indicates support.

Definition at line 168 of file relay.c.

References `client_print()`, `client_read_line()`, `log_pedantic`, `st_char_get()`, `st_length_get()`, and `st_length_int()`.

Referenced by `portal_smtp_relay_message()`, `smtp_forward_message()`, `smtp_relay_message()`, and `smtp_send_message()`.

5.165.1.22 int_t smtp_client_send_nullfrom (client_t * client)

Issue a MAIL FROM command to an smtp server with a null sender, and wait for a successful response.

Note:

Null senders are used when the sender is not concerned about being notified about bounced messages.

Parameters:

client a pointer to the network client to issue the MAIL FROM command.

Returns:

-2 if the remote server rejected the command, -1 on general network failure, or 1 on success.

Definition at line 196 of file relay.c.

References `client_print()`, `client_read_line()`, `log_pedantic`, `st_char_get()`, and `st_length_int()`.

Referenced by `smtp_bounce()`, and `smtp_reply()`.

5.165.1.23 int_t smtp_client_send_rcptto (client_t * *client*, stringer_t * *rcptto*)

Issue a RCPT TO command to an smtp server, and wait for a successful response.

Parameters:

client a pointer to the network client to issue the RCPT TO command.

rcptto a pointer to a managed string containing the recipient address parameter for the RCPT TO command.

Returns:

-2 if the remote server rejected the command, -1 on general network failure, or 1 on success.

Definition at line 219 of file relay.c.

References client_print(), client_read_line(), log_pedantic, st_char_get(), st_length_get(), and st_length_int().

Referenced by portal_smtp_relay_message(), smtp_bounce(), smtp_forward_message(), smtp_relay_message(), smtp_reply(), and smtp_send_message().

5.165.1.24 int_t smtp_compare (const void * *compare*, const void * *command*)

Definition at line 11 of file commands.c.

References command_t, PLACER, st_cmp_ci_eq(), and st_cmp_ci_starts().

Referenced by smtp_process(), and smtp_sort().

5.165.1.25 void smtp_data (connection_t * *con*)

Definition at line 1149 of file smtp.c.

References con_print(), con_write_bl(), magma, mail_add_required_headers(), mail_count_received(), mail_create_message(), mail_destroy_message(), mail_message_cleanup(), magma_t::relay_limit, requeue(), magma_t::smtp, smtp_data_inbound(), smtp_data_outbound(), smtp_data_read(), smtp_quit(), smtp_requeue(), and st_free().

Referenced by smtp_process().

5.165.1.26 void smtp_disabled (connection_t * *con*)

A stub function for an SMTP command that has not been implemented.

Note:

Executing a disabled command while result in a small delay and the protocol violation counter being incremented.

Returns:

This function returns no value.

Definition at line 163 of file smtp.c.

References con_print().

5.165.1.27 void smtp_ehlo (connection_t * *con*)

Process an SMTP EHLO command.

See also:

[smtp_parse_helo_domain\(\)](#)

Note:

Any prior domain specified by a HELO/EHLO command will be overwritten.

Parameters:

con the SMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 98 of file smtp.c.

References con_print(), con_secure(), con_write_bl(), magma, magma_t::message_length_limit, magma_t::smtp, smtp_parse_helo_domain(), st_char_get(), st_cleanup, and st_length_int().

5.165.1.28 int_t smtp_fetch_authorization (stringer_t * username, stringer_t * verification, smtp_outbound_prefs_t ** output)

Check if a user is authorized to send messages, and retrieve the user's outbound smtp preferences.

Note:

This function first checks if the account is locked in the Users table; next it populates the user's outbound mail preferences with a combination of data from the Users and Dispatch tables. The number of messages the user has sent in the past 24 hours is also computed from the Transmitting table.

Parameters:

cred a pointer to a credential object for the user, which must be of type CREDENTIAL_AUTH.

output a pointer to the address of an outbound smtp preferences object to receive the value of the lookup.

Returns:

0 on success or < 0 for failures, and > 0 for account locks. -2: general server or database error. -1: authentication failure (invalid username/password combination). 0: successful authentication

Definition at line 594 of file datatier.c.

References base64_encode_mod(), smtp_outbound_prefs_t::daily_send_limit, smtp_outbound_prefs_t::domain, smtp_outbound_prefs_t::importance, log_error, log_pedantic, mm_alloc(), mm_free(), mm_wipe(), res_field_int8(), res_field_string(), res_field_uint32(), res_field_uint64(), res_row_next(), res_table_free(), smtp_outbound_prefs_t::send_size_limit, smtp_outbound_prefs_t::sent_today, st_char_get(), st_free(), st_length_get(), st_length_int(), st_populated, stmt_get_result(), smtp_outbound_prefs_t::tls, and smtp_outbound_prefs_t::usernum.

Referenced by portal_outbound_checks(), smtp_auth_login(), and smtp_auth_plain().

5.165.1.29 stringer_t* smtp_fetch_autoreply (uint64_t autoreply, uint64_t usernum)

Fetch a specified auto-reply message for a user.

Note:

This function first checks the cache, and falls back to the database.

Parameters:

autoreply the numerical id of the auto-reply message in the database.

usernum the numerical id of the user to whom the auto-reply message belongs.

Returns:

NULL on failure, or a pointer to a managed string containing the user's auto-reply on success.

Definition at line 50 of file datatier.c.

References `cache_get()`, `cache_set()`, `log_pedantic`, `mm_wipe()`, `PLACER`, `res_field_string()`, `res_row_next()`, `res_table_free()`, and `stmt_get_result()`.

Referenced by `smtp_reply()`.

5.165.1.30 `int_t smtp_fetch_inbound (stringer_t * address, smtp_inbound_prefs_t ** output)`

Fetch a user's SMTP preferences for inbound mail.

Parameters:

cred a pointer to the credential object of a user with
address

Returns:

0 for success or < 0 for failures, and > 0 for account locks.

Return values:

-3,: for general server and database errors.
 -2,: if the domain isn't local.
 -1,: if the address is local but we didn't find a matching mailbox.
 0,: everything worked
 1,: the account is locked for inactivity (invalid username/password combination).

See also:

[AUTH_LOCK_INACTIVITY](#)

Return values:

2,: the account plan has expired.

See also:

[AUTH_LOCK_EXPIRED](#)

Return values:

3,: the account is subject to an administrative lock.

See also:

[AUTH_LOCK_ADMIN](#)

Return values:

4,: the account is locked due to suspicion of abuse violations.

See also:

[AUTH_LOCK_ABUSE](#)

Return values:

5,: the account has been locked at the request of the user.

See also:

[AUTH_LOCK_USER](#)

Definition at line 119 of file `datatier.c`.

References `smtp_inbound_filter_t::action`, `smtp_inbound_prefs_t::address`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_NONE`, `smtp_inbound_prefs_t::autoreply`, `base64_decode_mod()`, `BINARY`, `smtp_inbound_prefs_t::bounces`, `CONTIGUOUS`, `smtp_inbound_prefs_t::daily_rcv_limit`, `smtp_inbound_prefs_t::daily_rcv_limit_ip`, `smtp_inbound_prefs_t::dkim`, `smtp_inbound_prefs_t::dkimaction`, `smtp_inbound_prefs_t::domain`, `domain_wildcard()`, `smtp_inbound_filter_t::expression`, `smtp_inbound_filter_t::field`, `smtp_inbound_prefs_t::filters`, `smtp_inbound_filter_t::foldernum`, `smtp_inbound_prefs_t::forwarded`, `smtp_inbound_prefs_t::greylist`, `smtp_inbound_prefs_t::greytime`, `HEAP`, `smtp_inbound_prefs_t::inbox`, `inx_alloc()`, `inx_insert()`, `smtp_inbound_filter_t::label`, `smtp_inbound_filter_t::location`, `log_error`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `mail_domain_get()`, `MANAGED_T`, `mm_alloc()`, `mm_free()`, `mm_wipe()`, `NONE`, `smtp_inbound_prefs_t::overquota`, `smtp_inbound_prefs_t::phish`, `smtp_inbound_prefs_t::phishaction`, `PLACER`, `placer_t`, `prime_set()`, `smtp_inbound_prefs_t::quota`, `smtp_inbound_prefs_t::rbl`, `smtp_inbound_prefs_t::rblaction`, `smtp_inbound_prefs_t::rcptto`, `smtp_inbound_prefs_t::rcv_size_limit`, `res_field_block()`, `res_field_int8()`, `res_field_length()`, `res_field_string()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_count()`, `res_row_next()`, `res_table_free()`, `smtp_inbound_prefs_t::rollout`, `smtp_inbound_filter_t::rulenum`, `smtp_inbound_prefs_t::secure`, `smtp_inbound_prefs_t::signet`, `SMTP_FILTER_ACTION_LABEL`, `SMTP_FILTER_ACTION_MOVE`, `SMTP_FILTER_LOCATION_FIELD`, `smtp_free_inbound()`, `smtp_get_action()`, `smtp_list_free_filter()`, `smtp_inbound_prefs_t::spam`, `smtp_inbound_prefs_t::spamaction`, `smtp_inbound_prefs_t::spf`, `smtp_inbound_prefs_t::spfaction`, `st_char_get()`, `st_cleanup`, `st_dupe()`, `st_dupe_opts()`, `st_empty`, `st_length_get()`, `st_length_int()`, `stmt_get_result()`, `smtp_inbound_prefs_t::stor_size`, `smtp_inbound_filter_t::type`, `multi_t::u64`, `smtp_inbound_prefs_t::usenum`, `multi_t::val`, `smtp_inbound_prefs_t::virus`, and `smtp_inbound_prefs_t::virusaction`.

Referenced by `smtp_rcpt_to()`.

5.165.1.31 `table_t* smtp_fetch_rollmessages (uint64_t usenum)`

Retrieve a list, at a maximum of 20 entries, of the oldest messages owned by a user.

Parameters:

usenum the numerical id of the user whose messages are to be queried.

Returns:

NULL on failure, or a pointer to a sql results set containing the user's oldest messages on success.

Definition at line 345 of file `datatier.c`.

References `log_pedantic`, `mm_wipe()`, and `stmt_get_result()`.

Referenced by `smtp_rollout()`.

5.165.1.32 `int_t smtp_forward_message (server_t * server, stringer_t * sender, stringer_t * address, stringer_t * message, stringer_t * id, int_t mark, uint64_t signum, uint64_t sigkey)`

Definition at line 113 of file `transmit.c`.

References `client_t`, `HEAP`, `JOINTED`, `log_pedantic`, `mail_add_forward_headers()`, `MAPPED_T`, `PLACER`, `smtp_client_close()`, `smtp_client_connect()`, `smtp_client_send_data()`, `smtp_client_send_helo()`, `smtp_client_send_mailfrom()`, `smtp_client_send_rcptto()`, `st_dupe_opts()`, `st_free()`, `st_length_get()`, and `st_replace()`.

Referenced by `smtp_accept_message()`.

5.165.1.33 `void smtp_free_inbound (smtp_inbound_prefs_t * inbound)`

Free a list of SMTP inbound mail preferences and its underlying data.

Parameters:

inbound a pointer to the head of the SMTP inbound mail preferences list to be destroyed.

Returns:

This function returns no value.

Definition at line 228 of file session.c.

References smtp_inbound_prefs_t::address, smtp_inbound_prefs_t::domain, smtp_inbound_prefs_t::filters, smtp_inbound_prefs_t::forwarded, inx_cleanup(), mm_free(), smtp_inbound_prefs_t::next, prime_cleanup(), smtp_inbound_prefs_t::rcptto, smtp_inbound_prefs_t::signet, smtp_inbound_prefs_t::spamsig, and st_cleanup.

Referenced by smtp_fetch_inbound(), smtp_rcpt_to(), smtp_session_destroy(), and smtp_session_reset().

5.165.1.34 void smtp_free_outbound (smtp_outbound_prefs_t * *outbound*)

Free a set of SMTP outbound mail preferences and its underlying data.

Parameters:

outbound a pointer to the SMTP outbound mail preferences set to be destroyed.

Returns:

This function returns no value.

Definition at line 210 of file session.c.

References smtp_outbound_prefs_t::domain, mm_free(), smtp_outbound_prefs_t::recipients, smtp_free_recipients(), and st_cleanup.

Referenced by smtp_session_destroy().

5.165.1.35 void smtp_free_recipients (smtp_recipients_t * *recipients*)

Free a list of SMTP recipients and its underlying data.

Parameters:

recipients a pointer to the head of the recipients list to be destroyed.

Returns:

This function returns no value.

Definition at line 191 of file session.c.

References smtp_recipients_t::address, mm_free(), smtp_recipients_t::next, and st_cleanup.

Referenced by smtp_free_outbound(), and smtp_session_reset().

5.165.1.36 int_t smtp_get_action (chr_t * *string*, size_t *length*)

Parse an smtp event action string from the Dispatch table.

Note:

These values describe user-specified actions for events related to the spam filter, virus scanner, phishing detection, SPF, DKIM, and RBL checks.

Parameters:

string a pointer to a null-terminated string containing the name of the action.

length the length, in bytes, of the action string.

Returns:

the code of the corresponding smtp event action, SMTP_ACTION_UNDEFINED if the action is not known, or SMTP_ACTION_ERROR on failure.

Definition at line 18 of file `datatier.c`.

References `log_error`, `PLACER`, `SMTP_ACTION_BOUNCE`, `SMTP_ACTION_DELETE`, `SMTP_ACTION_ERROR`, `SMTP_ACTION_MARK`, `SMTP_ACTION_MARK_READ`, `SMTP_ACTION_REJECT`, `SMTP_ACTION_UNDEFINED`, and `st_cmp_cs_eq()`.

Referenced by `smtp_fetch_inbound()`.

5.165.1.37 void smtp_helo (connection_t * con)

Process an SMTP HELO command.

See also:

[smtp_parse_helo_domain\(\)](#)

Note:

Any prior domain specified by a HELO/EHLO command will be overwritten.

Parameters:

con the SMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 127 of file `smtp.c`.

References `con_print()`, `con_write_bl()`, `smtp_parse_helo_domain()`, `st_char_get()`, `st_cleanup`, and `st_length_int()`.

5.165.1.38 void smtp_init (connection_t * con)

The start of the protocol handler for the SMTP server.

Parameters:

con the new inbound SMTP client connection.

Returns:

This function returns no value.

Definition at line 1260 of file `smtp.c`.

References `con_print()`, `con_reverse_enqueue()`, `smtp_bypass_check()`, `smtp_requeue()`, `st_char_get()`, and `st_length_int()`.

Referenced by `protocol_enqueue()`, and `submission_init()`.

5.165.1.39 uint64_t smtp_insert_spamsig (smtp_inbound_prefs_t * prefs, uint64_t key, int_t code)

Store a spam signature in the database.

See also:

`dspam_process()`

Parameters:

prefs a pointer to the specified user's inbound mail preferences data.

key a randomly chosen authentication key for the signature.

code the dspam return code for the message from dspam_process()

Returns:

0 on failure, or the number of the newly inserted spam signature on success.

Definition at line 446 of file datatier.c.

References log_pedantic, mm_wipe(), smtp_inbound_prefs_t::spamsig, st_char_get(), st_length_get(), stmt_insert(), and smtp_inbound_prefs_t::usernum.

Referenced by smtp_store_spamsig().

5.165.1.40 void smtp_invalid (connection_t * con)

A function that is executed when an invalid SMTP command is executed.

Returns:

This function returns no value.

Definition at line 176 of file smtp.c.

References con_write_bl().

Referenced by smtp_process().

5.165.1.41 void smtp_list_free_filter (smtp_inbound_filter_t * filter)

Free an SMTP inbound filter and its underlying data.

Parameters:

filter a pointer to the SMTP inbound filter to be destroyed.

Returns:

This function returns no value.

Definition at line 249 of file session.c.

References smtp_inbound_filter_t::expression, smtp_inbound_filter_t::field, smtp_inbound_filter_t::label, mm_free(), and st_cleanup.

Referenced by smtp_fetch_inbound().

5.165.1.42 void smtp_mail_from (connection_t * con)

Specify the identity of a message's sender, in response to an SMTP MAIL FROM command.

See also:

[smtp_parse_mail_from_path\(\)](#)

Note:

This command must be preceded by a HELO command and successful authentication. Any prior email address specified by a MAIL FROM command will be overwritten. If the SIZE parameter was specified with the MAIL FROM command, its value will be compared to the maximum value specified in the smtp.message_length_limit configuration option.

Parameters:

con the SMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 57 of file smtp.c.

References `con_write_bl()`, `magma`, `magma_t::message_length_limit`, `magma_t::smtp`, `smtp_parse_mail_from_path()`, `smtp_session_reset()`, and `st_free()`.

5.165.1.43 void smtp_noop (connection_t * con)

Perform an SMTP NOOP (no-operation) command.

Note:

This command does essentially nothing and is mostly a way to keep connections alive without timing out due to inactivity.

Returns:

This function returns no value.

Definition at line 152 of file smtp.c.

References `con_write_bl()`.

5.165.1.44 stringer_t* smtp_parse_auth (stringer_t * data)

parse.c parse.c

Note:

This function will stop reading input when an invalid base64-encoding character is encountered.

Parameters:

con the SMTP client connection issuing the AUTH command.

Returns:

a managed string containing the domain specified by the AUTH command, or NULL on failure.

Definition at line 113 of file parse.c.

References `length`, `log_pedantic`, `st_empty_out()`, and `st_import()`.

Referenced by `smtp_auth_login()`, and `smtp_auth_plain()`.

5.165.1.45 stringer_t* smtp_parse_helo_domain (connection_t * con)

Parse the domain specified as the parameter to an SMTP HELO or EHLO command.

Note:

Valid domain names may only contain letters, numbers, periods and hyphens. This function will return a lowercase domain name, and stop reading input when an invalid character is encountered.

Parameters:

con the SMTP client connection issuing the command.

Returns:

a managed string containing the domain specified by the HELO command, or NULL on failure.

Definition at line 310 of file parse.c.

References magma_t::helo_length_limit, length, log_pedantic, lower_chr(), magma, pl_char_get(), pl_empty(), pl_length_get(), pl_length_int(), pl_trim_end(), magma_t::smtp, and st_import().

Referenced by smtp_ehlo(), and smtp_helo().

5.165.1.46 stringer_t* smtp_parse_mail_from_path (connection_t * con)

Extract the provided path from the command line Reverse-path = Path Forward-path = Path Path = "<" [A-d-l ":"] Mailbox ">" A-d-l = At-domain *("," A-d-l) At-domain = @ domain

Parameters:

con A connection structure which presumably contains a the MAIL FROM value inside the line buffer.

Returns:

Returns a stringer with the path that was extracted or NULL to indicate a problem.

Definition at line 168 of file parse.c.

References magma_t::address_length_limit, CONSTANT, length, log_pedantic, lower_chr(), magma, pl_char_get(), pl_empty(), pl_length_get(), pl_length_int(), pl_trim(), pl_trim_end(), PLACER, placer_t, magma_t::smtp, st_cmp_ci_starts(), st_cmp_cs_eq(), st_import(), tok_get_bl(), tok_get_count_bl(), tok_get_pl(), and uint64_conv_pl().

Referenced by smtp_mail_from().

5.165.1.47 stringer_t* smtp_parse_rcpt_to (connection_t * con)

LOW: The list of characters allowed in an email address needs to be verified against the RFC's. LOW: The parser should use different lists of valid characters for the the local and domain portions of an email address. Extract the provided path from the command line Reverse-path = Path Forward-path = Path Path = "<" [A-d-l ":"] Mailbox ">" A-d-l = At-domain *("," A-d-l) At-domain = @ domain

Parameters:

con A connection structure which presumably contains a the RCPT TO value inside the line buffer.

Returns:

Returns a stringer with the path that was extracted or NULL to indicate a problem.

Definition at line 24 of file parse.c.

References magma_t::address_length_limit, length, log_pedantic, lower_chr(), magma, pl_char_get(), pl_empty(), pl_length_get(), pl_length_int(), pl_trim_end(), magma_t::smtp, and st_import().

Referenced by smtp_rcpt_to().

5.165.1.48 void smtp_process (connection_t * con)

The main entry point in the SMTP server for processing commands issued by clients.

Parameters:

con a pointer to the connection object of the client issuing the SMTP command.

Returns:

This function returns no value.

Definition at line 48 of file commands.c.

References `command`, `command_t`, `con_read_line()`, `enqueue()`, `pl_char_get()`, `pl_empty()`, `pl_length_get()`, `requeue()`, `smtp_commands`, `smtp_compare()`, `smtp_data()`, `smtp_invalid()`, `smtp_process()`, `smtp_quit()`, and `smtp_requeue()`.

Referenced by `smtp_process()`, and `smtp_requeue()`.

5.165.1.49 void smtp_quit (connection_t * con)

Gracefully terminate an SMTP session, especially in response to an SMTP QUIT command.

Note:

The standards specify that the receiver **MUST** send an OK reply, and then close the transmission channel.

Parameters:

the SMTP client connection to be terminated.

Returns:

This function returns no value.

Definition at line 191 of file smtp.c.

References `con_destroy()`, `con_flush()`, `con_status()`, and `con_write_bl()`.

Referenced by `smtp_data()`, `smtp_process()`, and `smtp_requeue()`.

5.165.1.50 void smtp_rcpt_to (connection_t * con)

BUG: Detect messages 'from' a local user/domain and tell them to authenticate first.

Definition at line 550 of file smtp.c.

References `magma_t::abuse`, `magma_t::admin`, `AUTH_LOCK_ABUSE`, `AUTH_LOCK_ADMIN`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_INACTIVITY`, `AUTH_LOCK_USER`, `auth_sanitize_address()`, `con_addr()`, `con_addr_presentation()`, `con_print()`, `con_secure()`, `con_write_bl()`, `magma_t::contact`, `smtp_inbound_prefs_t::daily_rcv_limit`, `smtp_inbound_prefs_t::daily_rcv_limit_ip`, `smtp_inbound_prefs_t::forwarded`, `smtp_inbound_prefs_t::greylist`, `smtp_inbound_prefs_t::greytime`, `lower_st()`, `magma`, `mail_domain_get()`, `MANAGEDBUF`, `MEMORYBUF`, `smtp_inbound_prefs_t::overquota`, `pl_null()`, `placer_t`, `smtp_inbound_prefs_t::rbl`, `smtp_inbound_prefs_t::rblaction`, `smtp_inbound_prefs_t::rcptto`, `magma_t::recipient_limit`, `smtp_inbound_prefs_t::rcv_size_limit`, `smtp_inbound_prefs_t::rollout`, `magma_t::smtp`, `SMTP_ACTION_REJECT`, `smtp_add_inbound()`, `smtp_add_recipient()`, `smtp_check_duplicate_recipient()`, `smtp_check_greylist()`, `smtp_check_rbl()`, `smtp_check_receive_quota()`, `smtp_fetch_inbound()`, `smtp_free_inbound()`, `smtp_parse_rcpt_to()`, `smtp_inbound_prefs_t::spf`, `spf_check()`, `smtp_inbound_prefs_t::spfaction`, `st_char_get()`, `st_cmp_ci_eq()`, `st_free()`, `st_length_int()`, and `smtp_inbound_prefs_t::usernum`.

5.165.1.51 int_t smtp_relay_message (connection_t * con, stringer_t ** result)

Relay an outbound smtp message for a user.

Note:

The following process occurs before the message will be sent: 1. Necessary outbound headers are attached to the message.* 2. An outbound connection to a mail relay server is established (with a premium or normal server pool). 3. Once the connection is negotiated,

an RCPT TO command is issued for each of the message's recipients. 4. The mail message data is sent and the client connection is closed.

Parameters:

con a pointer to the connection object across which the outbound mail was attempted to be sent.

result a pointer to the address of a managed string that will receive the server's last response to the mail send attempt, regardless of whether or not it was successful.

Returns:

1 if the message was successfully sent or -1 on failure.

Definition at line 22 of file transmit.c.

References smtp_recipients_t::address, client_t, CONTIGUOUS, HEAP, log_pedantic, mail_add_outbound_headers(), MANAGED_T, smtp_recipients_t::next, PLACER, smtp_client_close(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), smtp_client_send_rcptto(), st_dupe_opts(), and st_replace().

Referenced by smtp_data_outbound().

5.165.1.52 int_t smtp_reply (stringer_t *from, stringer_t *to, uint64_t usernum, uint64_t autoreply, int_t spf, int_t dkim)

Definition at line 363 of file transmit.c.

References cache_get_u64(), cache_set_u64(), client_t, crc64_checksum(), dkim_signature_create(), lock_get(), lock_release(), log_pedantic, MANAGEDBUF, rand_choices(), smtp_client_close(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_nullfrom(), smtp_client_send_rcptto(), smtp_fetch_autoreply(), st_char_get(), st_cleanup, st_free(), st_length_int(), st_merge, and st_sprint().

Referenced by smtp_accept_message().

5.165.1.53 void smtp_requeue (connection_t *con)

commands.c

Definition at line 31 of file commands.c.

References con_status(), enqueue(), smtp_process(), smtp_quit(), and status.

Referenced by smtp_data(), smtp_init(), and smtp_process().

5.165.1.54 int_t smtp_rollout (smtp_inbound_prefs_t *prefs)

Delete the oldest mail message owned by a user until their storage usage falls below their storage quota.

Parameters:

prefs a pointer to the specified user's inbound mail preferences data.

Returns:

1 on success or < 0 on failure, where -1: An error occurred retrieving the rollout message list from the database. -2: The user lock could not be acquired.

Definition at line 79 of file accept.c.

References log_pedantic, mail_remove_message(), OBJECT_MESSAGES, smtp_inbound_prefs_t::quota, res_field_string(), res_field_uint32(), res_field_uint64(), res_row_next(), res_table_free(), serial_increment(), smtp_fetch_rollmessages(), st_char_get(), st_free(), smtp_inbound_prefs_t::stor_size, user_lock(), user_unlock(), and smtp_inbound_prefs_t::usernum.

Referenced by smtp_accept_message().

5.165.1.55 void smtp_rset (connection_t * con)

Reset the SMTP session, in response to an SMTP RSET command.

Note:

This command clears any sender, recipient, and mail data, along with all buffers and state tables. The connection structure is reset to the same it was in immediately after the HELO/EHLO command.

Parameters:

con the SMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 216 of file smtp.c.

References con_write_bl(), and smtp_session_reset().

5.165.1.56 int_t smtp_send_message (stringer_t * to, stringer_t * from, stringer_t * message)

Relay an outbound smtp message for the user.

Parameters:

to a managed string containing the name of the mail recipient.

from a managed string containing the address from which the email is being sent.

message a managed string containing the raw body of the mail message.

Returns:

-1 on error or 1 on success.

Definition at line 503 of file transmit.c.

References client_t, log_pedantic, smtp_client_close(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), and smtp_client_send_rcptto().

Referenced by contact_business(), and register_business_step2().

5.165.1.57 void smtp_session_destroy (connection_t * con)

Destroy the data associated with an SMTP session.

Parameters:

con the SMTP client connection to be destroyed.

Returns:

This function returns no value.

Definition at line 57 of file session.c.

References mail_destroy_message(), smtp_free_inbound(), smtp_free_outbound(), and st_cleanup.

Referenced by con_destroy().

5.165.1.58 void smtp_session_reset (connection_t * con)

Reset an SMTP session to its initialized state.

Parameters:

con the SMTP client connection to be reset.

Returns:

This function returns no value.

Definition at line 15 of file session.c.

References mail_destroy_message(), smtp_free_inbound(), smtp_free_recipients(), and st_cleanup.

Referenced by smtp_auth_login(), smtp_auth_plain(), smtp_data_inbound(), smtp_data_outbound(), smtp_mail_from(), smtp_rset(), and smtp_starttls().

5.165.1.59 void smtp_sort (void)

Sort the SMTP command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 26 of file commands.c.

References command_t, smtp_commands, and smtp_compare().

Referenced by protocol_init().

5.165.1.60 void smtp_starttls (connection_t * con)

TODO: Review error messages and update them with the appropriate response code. Initialize a TLS session for an unauthenticated SMTP session.

Parameters:

con the connection of the SMTP endpoint requesting the transport layer security upgrade.

Returns:

This function returns no value.

Definition at line 17 of file smtp.c.

References con_secure(), con_write_bl(), log_pedantic, M_SSL_BIO_NOCLOSE, pl_null(), smtp_session_reset(), st_length_set(), stats_increment_by_name(), and tls_server_alloc().

5.165.1.61 int_t smtp_store_message (smtp_inbound_prefs_t * prefs, stringer_t ** local)

Store a received SMTP message as a generic mail message, both on disk and in the database.

See also:

mail_store_messages()

Returns:

-1 on failure or 1 on success.

Definition at line 15 of file accept.c.

References `smtp_inbound_prefs_t::foldernum`, `log_error`, `log_pedantic`, `MAIL_MARK_BLACKHOLED`, `MAIL_MARK_INFECTED`, `MAIL_MARK_JUNK`, `MAIL_MARK_PHISHING`, `MAIL_MARK_SPOOFED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `mail_store_message()`, `smtp_inbound_prefs_t::mark`, `smtp_inbound_prefs_t::messagenum`, `OBJECT_MESSAGES`, `serial_increment()`, `smtp_inbound_prefs_t::signet`, `smtp_inbound_prefs_t::signum`, `SMTP_MARK_PHISH`, `SMTP_MARK_RBL`, `SMTP_MARK_READ`, `SMTP_MARK_SPAM`, `SMTP_MARK_SPOOF`, `SMTP_MARK_VIRUS`, `smtp_inbound_prefs_t::spamkey`, `status`, `user_lock()`, `user_unlock()`, and `smtp_inbound_prefs_t::usernum`.

Referenced by `smtp_accept_message()`.

5.165.1.62 `bool_t smtp_store_spamsig (smtp_inbound_prefs_t * prefs, int_t spam)`

Generate a random key for a spam signature and store it in the database.

Parameters:

prefs the user's smtp inbound preferences object, with the spam signature field set.

spam the dspam return code associated with the spam signature.

Returns:

true if the key was inserted into the database successfully, or false on failure.

Definition at line 161 of file accept.c.

References `imap_fetch_response_t::key`, `log_pedantic`, `rand_get_uint64()`, `smtp_inbound_prefs_t::signum`, `smtp_insert_spamsig()`, `smtp_inbound_prefs_t::spamkey`, and `uint64_digits()`.

Referenced by `smtp_accept_message()`.

5.165.1.63 `void smtp_update_receive_stats (connection_t * con, smtp_inbound_prefs_t * prefs)`

Update the receiving statistics and per-user log tables in the database for a successfully received smtp message.

Note:

The Receiving table is updated with the subnet address from which the message was received; the Log table for the user is updated to reflect the newly calculated totals of bounces or messages received.

Parameters:

con the connection across which the smtp message was received.

prefs a pointer to the user's smtp inbound mail preferences.

Returns:

This function returns no value.

Definition at line 374 of file datatier.c.

References `smtp_inbound_prefs_t::bounces`, `con_addr_subnet()`, `log_pedantic`, `mm_wipe()`, `PLACER`, `st_char_get()`, `st_cmp_cs_eq()`, `st_free()`, `st_length_get()`, `stmt_exec()`, and `smtp_inbound_prefs_t::usernum`.

Referenced by `smtp_accept_message()`.

5.165.1.64 `void smtp_update_transmission_stats (connection_t * con)`

Update the transmission and per-user log tables in the database for a successfully sent smtp message.

Note:

The Transmitting table is updated with the timestamp of this transaction; the Log table for the user is updated to reflect the newly calculated total for messages sent.

Parameters:

con a pointer to the connection object across which the smtp message was sent.

prefs a pointer to the user's smtp inbound mail preferences.

Returns:

This function returns no value.

Definition at line 498 of file datatier.c.

References log_pedantic, mm_wipe(), and stmt_exec().

Referenced by smtp_data_outbound().

5.165.1.65 void submission_init (connection_t * *con*)

Definition at line 1277 of file smtp.c.

References smtp_init().

Referenced by protocol_enqueue().

5.166 src/network/write.c File Reference

```
#include "magma.h"
```

Functions

- `int64_t con_write_bl (connection_t *con, char *block, size_t length)`
Write data to a network connection.
- `int64_t con_write_st (connection_t *con, stringer_t *string)`
Write a managed string to a network connection.
- `int64_t con_write_ns (connection_t *con, char *string)`
Write a null-terminated string to a network connection.
- `int64_t con_write_pl (connection_t *con, placer_t string)`
Write a placer to a network connection.
- `int64_t con_print (connection_t *con, chr_t *format,...)`
Write a formatted string to a network connection.
- `int64_t client_write (client_t *client, stringer_t *s)`
Write data to a network client.
- `int64_t client_print (client_t *client, chr_t *format,...)`
Write a formatted string to a network client.

5.166.1 Function Documentation

5.166.1.1 `int64_t client_print (client_t *client, chr_t *format, ...)`

Write a formatted string to a network client. [write.c](#)

See also:

[client_write\(\)](#)

Parameters:

- client*** the network client connection to which the supplied data will be written.
- format*** the format string for the data to be written to the network client connection.
- ...*** a va_arg style collection of variables to be expanded by the passed format string.

Returns:

- 1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 258 of file [write.c](#).

References [client_write\(\)](#), [st_free\(\)](#), and [st_vaprint](#).

Referenced by [smtp_client_send_helo\(\)](#), [smtp_client_send_mailfrom\(\)](#), [smtp_client_send_nullfrom\(\)](#), and [smtp_client_send_rcptto\(\)](#).

5.166.1.2 `int64_t client_write (client_t * client, stringer_t * s)`

Write data to a network client.

Note:

This function works regardless of whether or not the client connection is ssl-enabled. If the network write requires multiple system calls, then this code will loop until all the data has been transmitted.

Parameters:

client the network client connection to which the supplied data will be written.
s a managed string containing the data to be written to the network client connection.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 169 of file write.c.

References `bytes`, `client_status()`, `ip_presentation()`, `length`, `log_pedantic`, `MANAGEDDBUF`, `MEMORYBUF`, `st_char_get()`, `st_empty_out()`, `st_length_int()`, `status`, `tcp_status()`, `tls_cipher()`, `tls_error()`, and `tls_write()`.

Referenced by `client_print()`, `smtp_client_close()`, and `smtp_client_send_data()`.

5.166.1.3 `int64_t con_print (connection_t * con, chr_t * format, ...)`

Write a formatted string to a network connection.

See also:

[con_write_bl\(\)](#)

Parameters:

con the connection across which the supplied data will be written.
format a format string for the output string to be written to the connection's remote client.
 ... a variable argument list of parameters for the specified format string.

Returns:

-1 on general failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 112 of file write.c.

References `buflen`, `bufptr`, `bytes`, `con_write_bl()`, `length`, `mm_alloc()`, and `mm_free()`.

Referenced by `dmtpl_ehlo()`, `dmtpl_helo()`, `dmtpl_init()`, `http_parse_header()`, `http_print_301()`, `http_response_header()`, `http_response_options()`, `imap_append()`, `imap_capability()`, `imap_check()`, `imap_close()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_id()`, `imap_idle()`, `imap_init()`, `imap_invalid()`, `imap_list()`, `imap_login()`, `imap_logout()`, `imap_lsub()`, `imap_noop()`, `imap_process()`, `imap_rename()`, `imap_search()`, `imap_select()`, `imap_starttls()`, `imap_status()`, `imap_store()`, `imap_subscribe()`, `imap_unsubscribe()`, `molten_stats()`, `pop_capa()`, `pop_last()`, `pop_list()`, `pop_retr()`, `pop_stat()`, `pop_top()`, `pop_uidl()`, `register_print_captcha()`, `smtp_data()`, `smtp_data_inbound()`, `smtp_data_outbound()`, `smtp_disabled()`, `smtp_ehlo()`, `smtp_helo()`, `smtp_init()`, `smtp_rcpt_to()`, and `teacher_print_message()`.

5.166.1.4 `int64_t con_write_bl (connection_t * con, char * block, size_t length)`

Write data to a network connection. **HIGH:** Create a simpler method of triggering a queue event following a connection write operation, and audit the code to ensure write calls do not accidentally orphan a connection by not queuing the connection upon completion. In other words, always ensure that [enqueue\(\)](#) is being called on a connection after all processing is performed, so that it is not lost (whether it is to be kept or not).

Note:

This function works regardless of whether or not the connection is ssl-enabled. If the network write requires multiple system calls, then this code will loop until all the data has been transmitted.

Parameters:

con the connection across which the supplied data will be written.

block a pointer to a data buffer containing the data to be written to the connection's remote client.

length the length, in bytes, of the data buffer to be written.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 24 of file write.c.

References bytes, con_status(), status, tcp_write(), and tls_write().

Referenced by con_print(), con_write_ns(), con_write_pl(), con_write_st(), dmtpl_data(), dmtpl_help(), dmtpl_hist(), dmtpl_invalid(), dmtpl_mail(), dmtpl_mode(), dmtpl_noop(), dmtpl_quit(), dmtpl_rcpt(), dmtpl_rset(), dmtpl_sgnt(), dmtpl_verb(), dmtpl_vrfy(), imap_fetch(), imap_logout(), imap_parse_literal(), imap_process(), molten_invalid(), molten_stats(), pop_delete(), pop_init(), pop_invalid(), pop_list(), pop_noop(), pop_pass(), pop_quit(), pop_retr(), pop_rset(), pop_starttls(), pop_top(), pop_uidl(), pop_user(), smtp_auth_login(), smtp_auth_plain(), smtp_data(), smtp_data_inbound(), smtp_data_outbound(), smtp_ehlo(), smtp_helo(), smtp_invalid(), smtp_mail_from(), smtp_noop(), smtp_quit(), smtp_rcpt_to(), smtp_rset(), and smtp_starttls().

5.166.1.5 int64_t con_write_ns (connection_t * con, char * string)

Write a null-terminated string to a network connection.

See also:

[con_write_bl\(\)](#)

Parameters:

con the connection across which the supplied data will be written.

string a pointer to a null-terminated string containing the data to be written to the connection's remote client.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 89 of file write.c.

References con_write_bl(), and ns_length_get().

5.166.1.6 int64_t con_write_pl (connection_t * con, placer_t string)

Write a placer to a network connection.

See also:

[con_write_bl\(\)](#)

Parameters:

con the connection across which the supplied data will be written.

string a placer pointing to the data to be written to the connection's remote client.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 100 of file write.c.

References `con_write_bl()`, `pl_char_get()`, and `pl_length_get()`.

5.166.1.7 int64_t con_write_st (connection_t * *con*, stringer_t * *string*)

Write a managed string to a network connection.

See also:

[con_write_bl\(\)](#)

Parameters:

con the connection across which the supplied data will be written.

string a managed string containing the data to be written to the connection's remote client.

Returns:

-1 on general network failure, -2 if the connection was reset or closed, or the number of bytes that were written across the connection.

Definition at line 78 of file write.c.

References `con_write_bl()`, `st_char_get()`, and `st_length_get()`.

Referenced by `api_response()`, `contact_print_form()`, `contact_print_message()`, `http_print_400()`, `http_print_403()`, `http_print_404()`, `http_print_405()`, `http_print_500()`, `http_print_500_log()`, `http_print_501()`, `http_response()`, `imap_fetch()`, `imap_search()`, `pop_retr()`, `pop_top()`, `portal_endpoint_attachments_progress()`, `portal_endpoint_error()`, `portal_endpoint_response()`, `portal_endpoint_search()`, `portal_print_login()`, `register_print_captcha()`, `register_print_message()`, `register_print_step1()`, `register_print_step2()`, `register_print_step3()`, `smtp_data_outbound()`, `statistics_process()`, `teacher_print_form()`, and `teacher_print_message()`.

5.167 src/objects/auth/auth.c File Reference

```
#include "magma.h"
```

Functions

- `void auth_free (auth_t *auth)`
Free an authentication object.
- `auth_t * auth_alloc (void)`
Allocate an empty authentication object.
- `auth_t * auth_challenge (stringer_t *username)`
Used to sanitize and normalize a username, then retrieve the authentication information for that account. For STACIE authentications, this function creates a nonce value, which the user's client can combine with the verification token value to derive an ephemeral login token, which may only be used once.
- `int_t auth_response (auth_t *auth, stringer_t *ephemeral)`
Test an ephemeral token for validity. If the token is invalid, generate a different nonce for the next attempt.
- `int_t auth_login (stringer_t *username, stringer_t *password, auth_t **output)`
Loads a user's authentication information and calculates the tokens for comparison.

5.167.1 Function Documentation

5.167.1.1 auth_t* auth_alloc (void)

Allocate an empty authentication object. [auth.c](#)

Returns:

an empty authentication object allocated off the stack.

Definition at line 32 of file [auth.c](#).

References [log_pedantic](#), [mm_alloc\(\)](#), and [mm_wipe\(\)](#).

Referenced by [auth_challenge\(\)](#).

5.167.1.2 auth_t* auth_challenge (stringer_t * username)

Used to sanitize and normalize a username, then retrieve the authentication information for that account. For STACIE authentications, this function creates a nonce value, which the user's client can combine with the verification token value to derive an ephemeral login token, which may only be used once.

Parameters:

username the unsanitized username, which may also be an email address.

Returns:

Definition at line 54 of file auth.c.

References `auth_alloc()`, `auth_data_fetch()`, `auth_free()`, `auth_sanitise_username()`, `log_error`, `log_pedantic`, `auth_t::nonce`, `auth_t::seasoning`, `st_empty`, `st_length_get()`, `stacie_create_nonce()`, `STACIE_NONCE_LENGTH`, `auth_t::tokens`, `auth_t::username`, and `auth_t::verification`.

Referenced by `auth_login()`, and `teacher_process()`.

5.167.1.3 `void auth_free (auth_t * auth)`

Free an authentication object.

Parameters:

auth a pointer to the authentication object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file auth.c.

References `auth_t::ephemeral`, `auth_t::key`, `auth_t::keys`, `auth_t::legacy`, `log_pedantic`, `auth_t::master`, `mm_free()`, `auth_t::nonce`, `auth_t::salt`, `auth_t::seasoning`, `st_cleanup`, `auth_t::token`, `auth_t::tokens`, `auth_t::username`, and `auth_t::verification`.

Referenced by `api_endpoint_auth()`, `auth_challenge()`, `auth_login()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, `smtp_auth_plain()`, and `teacher_process()`.

5.167.1.4 `int_t auth_login (stringer_t * username, stringer_t * password, auth_t ** output)`

Loads a user's authentication information and calculates the tokens for comparison.

Note:

If the username and password combination is valid, but involves an account with legacy authentication tokens, the plain text password is used to replace the legacy tokens with STACIE compliant values. Also note the complexity of this function. It will be far simpler once support for legacy login tokens is removed.

Parameters:

username the unsanitized username, provided with the login attempt.

password the plain text password.

output pointer location which will hold the resulting `auth_t` structure, but only if the inputs authenticate successfully, and 0 is returned; also note that output must be pointing at NULL when its passed in to avoid creating a memory leak.

Returns:

if a technical error occurs unrelated to the provided credentials, -1 will be returned, if the username and password combination are invalid, 1 will be returned. The output parameter is invalid, 1 will be returned, and the output will be emptied, a 0 is returned when login is successful.

Definition at line 177 of file auth.c.

References `auth_challenge()`, `auth_data_update_legacy()`, `auth_data_update_lock()`, `auth_free()`, `auth_legacy()`, `auth_legacy_free()`, `AUTH_LOCK_INACTIVITY`, `AUTH_LOCK_NONE`, `auth_stacie()`, `auth_stacie_free()`, `base64_encode_mod()`, `auth_t::bonus`, `hex_encode_st()`, `auth_legacy_t::key`, `auth_t::key`, `auth_stacie_t::keys`, `auth_t::keys`, `auth_t::legacy`, `auth_t::locked`, `log_error`, `log_pedantic`, `magma`, `auth_stacie_t::master`, `auth_t::master`, `magma_t::minimum_password_length`, `rand_get_uint32()`, `auth_t::salt`, `auth_t::seasoning`, `magma_t::secure`, `st_char_get()`, `st_cleanup`, `st_cmp_cs_eq()`, `st_dupe()`, `st_empty`, `st_length_get()`, `st_length_int()`, `stacie_create_salt()`, `auth_t::status`, `auth_legacy_t::token`, `auth_t::token`, `auth_stacie_t::tokens`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, `utf8_length_st()`, `auth_stacie_t::verification`, and `auth_t::verification`.

Referenced by `api_endpoint_auth()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, and `smtp_auth_plain()`.

5.167.1.5 `int_t auth_response (auth_t * auth, stringer_t * ephemeral)`

Test an ephemeral token for validity. If the token is invalid, generate a different nonce for the next attempt.

Parameters:

auth the challenge values, including the verification token, and a nonce value.

ephemeral the ephemeral token value provided by the user for comparison.

Returns:

return -1 if an error occurs, 0 if the response is validated, and 1 if the ephemeral token is invalid.

Definition at line 98 of file auth.c.

References `auth_stacie()`, `auth_stacie_cleanup()`, `auth_stacie_free()`, `auth_t::bonus`, `auth_t::ephemeral`, `auth_stacie_t::ephemeral`, `log_pedantic`, `mm_move()`, `auth_t::nonce`, `auth_t::salt`, `auth_t::seasoning`, `st_char_get()`, `st_cmp_cs_eq()`, `st_data_get()`, `st_dupe()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_int()`, `stacie_create_nonce()`, `STACIE_NONCE_LENGTH`, `STACIE_TOKEN_LENGTH`, `auth_stacie_t::tokens`, `auth_t::tokens`, `auth_t::username`, and `auth_t::verification`.

5.168 src/objects/auth/auth.h File Reference

Data Structures

- struct [auth_legacy_t](#)
- struct [auth_stacie_t](#)
- struct [auth_t](#)

Enumerations

- enum [auth_lock_status_t](#) {
[AUTH_LOCK_NONE](#) = 0, [AUTH_LOCK_INACTIVITY](#) = 1, [AUTH_LOCK_EXPIRED](#) = 2, [AUTH_LOCK_ADMIN](#) = 3,
[AUTH_LOCK_ABUSE](#) = 4, [AUTH_LOCK_USER](#) = 5 }

Functions

- [stringer_t * auth_sanitize_address](#) ([stringer_t](#) *username)
[username.c](#)
- [stringer_t * auth_sanitize_username](#) ([stringer_t](#) *username)
Get the username portion of a fully qualified email address.
- [auth_t * auth_alloc](#) (void)
[auth.c](#)
- [auth_t * auth_challenge](#) ([stringer_t](#) *username)
Used to sanitize and normalize a username, then retrieve the authentication information for that account. For STACIE authentications, this function creates a nonce value, which the user's client can combine with the verification token value to derive an ephemeral login token, which may only be used once.
- void [auth_free](#) ([auth_t](#) *auth)
Free an authentication object.
- [int_t auth_login](#) ([stringer_t](#) *username, [stringer_t](#) *password, [auth_t](#) **output)
Loads a user's authentication information and calculates the tokens for comparison.
- [int_t auth_response](#) ([auth_t](#) *auth, [stringer_t](#) *ephemeral)
Test an ephemeral token for validity. If the token is invalid, generate a different nonce for the next attempt.
- [int_t auth_data_fetch](#) ([auth_t](#) *auth)
[datatier.c](#)
- [int_t auth_data_update_legacy](#) (uint64_t usernum, [stringer_t](#) *legacy, [stringer_t](#) *salt, [stringer_t](#) *verification, uint32_t bonus)
Replaces legacy auth tokens with STACIE compatible tokens.
- void [auth_data_update_lock](#) (uint64_t usernum, [auth_lock_status_t](#) lock)
Update a user lock status in the database.
- [auth_stacie_t * auth_stacie](#) (uint32_t bonus, [stringer_t](#) *username, [stringer_t](#) *password, [stringer_t](#) *salt, [stringer_t](#) *verification, [stringer_t](#) *nonce)
[stacie.c](#)

- [auth_stacie_t * auth_stacie_alloc](#) (void)
Allocate and initialize a STACIE object.
- void [auth_stacie_cleanup](#) ([auth_stacie_t](#) *stacie)
A checked cleanup function which can be used free a STACIE object.
- void [auth_stacie_free](#) ([auth_stacie_t](#) *stacie)
A checked cleanup function which can be used free a STACIE object.
- [auth_legacy_t * auth_legacy](#) ([stringer_t](#) *username, [stringer_t](#) *password)
legacy.c
- [auth_legacy_t * auth_legacy_alloc](#) (void)
- void [auth_legacy_free](#) ([auth_legacy_t](#) *legacy)

5.168.1 Enumeration Type Documentation

5.168.1.1 enum auth_lock_status_t

Enumerator:

- AUTH_LOCK_NONE** Normal unlocked account.
- AUTH_LOCK_INACTIVITY** An inactive account lock, that gets cleared automatically when the user authenticates.
- AUTH_LOCK_EXPIRED** The user subscription has expired, messages are accepted, but logins are refused.
- AUTH_LOCK_ADMIN** An administrative lock, and used for a variety of reasons.
- AUTH_LOCK_ABUSE** Abuse locks are applied to accounts that appear to be violating the acceptable use policy.
- AUTH_LOCK_USER** The account has been locked at the request of the user.

Definition at line 14 of file auth.h.

5.168.2 Function Documentation

5.168.2.1 auth_t* auth_alloc (void)

[auth.c](#) [auth.c](#)

Returns:

an empty authentication object allocated off the stack.

Definition at line 32 of file auth.c.

References [log_pedantic](#), [mm_alloc\(\)](#), and [mm_wipe\(\)](#).

Referenced by [auth_challenge\(\)](#).

5.168.2.2 auth_t* auth_challenge (stringer_t * username)

Used to sanitize and normalize a username, then retrieve the authentication information for that account. For STACIE authentications, this function creates a nonce value, which the user's client can combine with the verification token value to derive an ephemeral login token, which may only be used once.

Parameters:

username the unsanitized username, which may also be an email address.

Returns:

Definition at line 54 of file auth.c.

References `auth_alloc()`, `auth_data_fetch()`, `auth_free()`, `auth_sanitize_username()`, `log_error`, `log_pedantic`, `auth_t::nonce`, `auth_t::seasoning`, `st_empty`, `st_length_get()`, `stacie_create_nonce()`, `STACIE_NONCE_LENGTH`, `auth_t::tokens`, `auth_t::username`, and `auth_t::verification`.

Referenced by `auth_login()`, and `teacher_process()`.

5.168.2.3 int_t auth_data_fetch (auth_t * auth)

datatier.c datatier.c

Note:

This function searches the Users table first, and the Mailboxes table second. This way if someone sneaks a username into the Mailboxes table, they won't override the Users table.

Parameters:

auth The authentication object providing the username, and used to store the result.

Returns:

0 if the user information is pulled correctly, 1 if the user isn't found, and -1 if an error occurs.

Definition at line 134 of file datatier.c.

References `base64_decode_mod()`, `auth_t::bonus`, `hex_decode_st()`, `auth_t::legacy`, `auth_t::locked`, `log_error`, `log_pedantic`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_int8()`, `res_field_length()`, `res_field_string()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_count()`, `res_row_next()`, `res_table_free()`, `auth_t::salt`, `auth_t::seasoning`, `st_char_get()`, `st_cmp_cs_eq()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_int()`, `st_search_chr()`, `STACIE_KEY_ROUNDS_MAX`, `STACIE_SALT_LENGTH`, `STACIE_TOKEN_LENGTH`, `auth_t::status`, `stmt_get_result()`, `auth_t::token`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, and `auth_t::verification`.

Referenced by `auth_challenge()`.

5.168.2.4 int_t auth_data_update_legacy (uint64_t usernum, stringer_t * legacy, stringer_t * salt, stringer_t * verification, uint32_t bonus)

Replaces legacy auth tokens with STACIE compatible tokens.

Parameters:

usernum The user account number.

legacy The legacy auth token, encoded as a hexadecimal string.

salt The user specific salt value, encoded as a hexadecimal string.

verification The STACIE verification token, encoded as a hexadecimal string.

bonus The number of bonus hash rounds.

Returns:

0 if the user information was updated correctly, or -1 if an error occurs.

Definition at line 22 of file datatier.c.

References `log_error`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_empty`, `st_length_get()`, `STACIE_KEY_ROUNDS_MAX`, and `stmt_exec_affected()`.

Referenced by `auth_login()`.

5.168.2.5 void auth_data_update_lock (uint64_t *usernum*, auth_lock_status_t *lock*)

Update a user lock status in the database.

Parameters:

usernum the numerical id of the user for whom the lock will be set.

lock the new value to which the specified user's lock will be set.

Returns:

This function returns no value.

Definition at line 89 of file `datatier.c`.

References `log_pedantic`, `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `auth_login()`.

5.168.2.6 void auth_free (auth_t * *auth*)

Free an authentication object.

Parameters:

auth a pointer to the authentication object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file `auth.c`.

References `auth_t::ephemeral`, `auth_t::key`, `auth_t::keys`, `auth_t::legacy`, `log_pedantic`, `auth_t::master`, `mm_free()`, `auth_t::nonce`, `auth_t::salt`, `auth_t::seasoning`, `st_cleanup`, `auth_t::token`, `auth_t::tokens`, `auth_t::username`, and `auth_t::verification`.

Referenced by `api_endpoint_auth()`, `auth_challenge()`, `auth_login()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, `smtp_auth_plain()`, and `teacher_process()`.

5.168.2.7 auth_legacy_t* auth_legacy (stringer_t * *username*, stringer_t * *password*)

[legacy.c](#) `legacy.c`

Parameters:

username a managed string holding the sanitized username.

password a managed string holding the plain text user password.

Returns:

an [auth_legacy_t](#) structure is returned upon success, and NULL upon failure.

Definition at line 60 of file `legacy.c`.

References `auth_legacy_alloc()`, `auth_legacy_free()`, `CONTIGUOUS`, `hash_sha512()`, `auth_legacy_t::key`, `log_error`, `magma`, `MANAGED_T`, `MANAGEDBUF`, `magma_t::salt`, `SECURE`, `magma_t::secure`, `st_alloc()`, `st_alloc_opts()`, `st_empty`, `st_free()`, `st_merge`, and `auth_legacy_t::token`.

Referenced by `auth_login()`.

5.168.2.8 `auth_legacy_t* auth_legacy_alloc (void)`

Definition at line 43 of file legacy.c.

References `mm_alloc()`, and `mm_wipe()`.

Referenced by `auth_legacy()`.

5.168.2.9 `void auth_legacy_free (auth_legacy_t * legacy)`

Definition at line 34 of file legacy.c.

References `auth_legacy_t::key`, `mm_free()`, `st_cleanup`, and `auth_legacy_t::token`.

Referenced by `auth_legacy()`, and `auth_login()`.

5.168.2.10 `int_t auth_login (stringer_t * username, stringer_t * password, auth_t ** output)`

Loads a user's authentication information and calculates the tokens for comparison.

Note:

If the username and password combination is valid, but involves an account with legacy authentication tokens, the plain text password is used to replace the legacy tokens with STACIE compliant values. Also note the complexity of this function. It will be far simpler once support for legacy login tokens is removed.

Parameters:

username the unsanitized username, provided with the login attempt.

password the plain text password.

output pointer location which will hold the resulting `auth_t` structure, but only if the inputs authenticate successfully, and 0 is returned; also note that output must be pointing at NULL when its passed in to avoid creating a memory leak.

Returns:

if a technical error occurs unrelated to the provided credentials, -1 will be returned, if the username and password combination are invalid, 1 will be returned. The output parameter is invalid, 1 will be returned, and the output will be emptied, a 0 is returned when login is successful.

Definition at line 177 of file auth.c.

References `auth_challenge()`, `auth_data_update_legacy()`, `auth_data_update_lock()`, `auth_free()`, `auth_legacy()`, `auth_legacy_free()`, `AUTH_LOCK_INACTIVITY`, `AUTH_LOCK_NONE`, `auth_stacie()`, `auth_stacie_free()`, `base64_encode_mod()`, `auth_t::bonus`, `hex_encode_st()`, `auth_legacy_t::key`, `auth_t::key`, `auth_stacie_t::keys`, `auth_t::keys`, `auth_t::legacy`, `auth_t::locked`, `log_error`, `log_pedantic`, `magma`, `auth_stacie_t::master`, `auth_t::master`, `magma_t::minimum_password_length`, `rand_get_uint32()`, `auth_t::salt`, `auth_t::seasoning`, `magma_t::secure`, `st_char_get()`, `st_cleanup`, `st_cmp_cs_eq()`, `st_dupe()`, `st_empty`, `st_length_get()`, `st_length_int()`, `stacie_create_salt()`, `auth_t::status`, `auth_legacy_t::token`, `auth_t::token`, `auth_stacie_t::tokens`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, `utf8_length_st()`, `auth_stacie_t::verification`, and `auth_t::verification`.

Referenced by `api_endpoint_auth()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, and `smtp_auth_plain()`.

5.168.2.11 `int_t auth_response (auth_t * auth, stringer_t * ephemeral)`

Test an ephemeral token for validity. If the token is invalid, generate a different nonce for the next attempt.

Parameters:

auth the challenge values, including the verification token, and a nonce value.

ephemeral the ephemeral token value provided by the user for comparison.

Returns:

return -1 if an error occurs, 0 if the response is validated, and 1 if the ephemeral token is invalid.

Definition at line 98 of file auth.c.

References `auth_stacie()`, `auth_stacie_cleanup()`, `auth_stacie_free()`, `auth_t::bonus`, `auth_t::ephemeral`, `auth_stacie_t::ephemeral`, `log_pedantic`, `mm_move()`, `auth_t::nonce`, `auth_t::salt`, `auth_t::seasoning`, `st_char_get()`, `st_cmp_cs_eq()`, `st_data_get()`, `st_dupe()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_int()`, `stacie_create_nonce()`, `STACIE_NONCE_LENGTH`, `STACIE_TOKEN_LENGTH`, `auth_stacie_t::tokens`, `auth_t::tokens`, `auth_t::username`, and `auth_t::verification`.

5.168.2.12 stringer_t* auth_sanitize_address (stringer_t * username)

[username.c](#) [username.c](#)

Parameters:

username a managed string containing the user supplied login name to be processed, it may contain an optional domain suffix.

Note:

This function duplicates the input address string, with all characters converted to lowercase, and whitespace removed. '.' and '-' are also converted to '_' in the username, and if there is a '+' in the local portion of an email address, all subsequent characters in that local part will be ignored, as they constitute a label.

Returns:

NULL on failure, or a managed string containing the sanitized address on success, that must be freed by the caller.

Definition at line 18 of file username.c.

References `chr_whitespace()`, `lower_chr()`, `PLACER`, `st_alloc()`, `st_char_get()`, `st_empty_out()`, `st_length_get()`, and `st_length_set()`.

Referenced by `auth_sanitize_username()`, `smtp_data_outbound()`, and `smtp_rcpt_to()`.

5.168.2.13 stringer_t* auth_sanitize_username (stringer_t * username)

Get the username portion of a fully qualified email address.

Parameters:

username a managed string containing the username to be processed, with an optional domain suffix.

Returns:

NULL on failure, or a managed string containing the credential username portion of the supplied address.

Definition at line 103 of file username.c.

References `auth_sanitize_address()`, `magma_t::domain`, `magma`, `PLACER`, `st_char_get()`, `st_cmp_cs_eq()`, `st_length_get()`, `st_length_set()`, `st_search_cs()`, and `magma_t::system`.

Referenced by `api_endpoint_register()`, and `auth_challenge()`.

5.168.2.14 auth_stacie_t* auth_stacie (uint32_t bonus, stringer_t * username, stringer_t * password, stringer_t * salt, stringer_t * verification, stringer_t * nonce)

[stacie.c](#) [stacie.c](#)

Note:

If the password is NULL, then a nonce is required, and only the ephemeral token will be populated in the resulting [auth_stacie_t](#) structure. Likewise, if the nonce is NULL, then the password must be provided, and the ephemeral token will be absent in the resulting [auth_stacie_t](#) structure. If the password and nonce are both set to NULL, then an error is triggered and NULL is returned.

Parameters:

bonus the number of bonus hash rounds to be applied for this particular user account.

username a managed string holding the normalized username.

password a managed string holding the plain text user password.

salt a managed string with the salt value for the current user.

nonce the randomly generated nonce value for the ephemeral token.

Returns:

an [auth_stacie_t](#) structure is returned upon success, and NULL upon failure.

Definition at line 92 of file stacie.c.

References [auth_stacie_alloc\(\)](#), [auth_stacie_free\(\)](#), [auth_stacie_t::ephemeral](#), [auth_stacie_t::keys](#), [log_pedantic](#), [auth_stacie_t::master](#), [auth_stacie_t::password](#), [st_empty](#), [stacie_derive_key\(\)](#), [stacie_derive_rounds\(\)](#), [stacie_derive_seed\(\)](#), [stacie_derive_token\(\)](#), [STACIE_KEY_ROUNDS_MAX](#), [auth_stacie_t::tokens](#), and [auth_stacie_t::verification](#).

Referenced by [auth_login\(\)](#), [auth_response\(\)](#), and [register_data_insert_user\(\)](#).

5.168.2.15 [auth_stacie_t*](#) [auth_stacie_alloc](#) (void)

Allocate and initialize a STACIE object.

Returns:

NULL on failure, or a pointer to the newly allocated STACIE object on success.

Definition at line 60 of file stacie.c.

References [log_pedantic](#), [mm_alloc\(\)](#), and [mm_wipe\(\)](#).

Referenced by [auth_stacie\(\)](#).

5.168.2.16 [void](#) [auth_stacie_cleanup](#) ([auth_stacie_t](#) * *stacie*)

A checked cleanup function which can be used free a STACIE object.

Note:

Use this function when the STACIE object pointer could potentially be NULL. Call the free function directly if the STACIE object is guaranteed not to be NULL.

See also:

[auth_stacie_free\(\)](#)

Parameters:

stacie the STACIE object which will be freed.

Returns:

This function returns no value.

Definition at line 48 of file stacie.c.

References `auth_stacie_free()`.

Referenced by `auth_response()`, and `register_data_insert_user()`.

5.168.2.17 void auth_stacie_free (auth_stacie_t * *stacie*)

A checked cleanup function which can be used free a STACIE object.

Note:

Use this function when the STACIE object is guaranteed not to be NULL. Call the cleanup function directly if the STACIE object pointer could potentially be NULL.

See also:

[auth_stacie_cleanup\(\)](#)

Parameters:

stacie the STACIE object which will be freed.

Returns:

This function returns no value.

Definition at line 19 of file stacie.c.

References `auth_stacie_t::ephemeral`, `auth_stacie_t::keys`, `log_pedantic`, `auth_stacie_t::master`, `mm_free()`, `auth_stacie_t::password`, `st_cleanup`, `auth_stacie_t::tokens`, and `auth_stacie_t::verification`.

Referenced by `auth_login()`, `auth_response()`, `auth_stacie()`, and `auth_stacie_cleanup()`.

5.169 src/objects/auth/legacy.c File Reference

```
#include "magma.h"
```

Functions

- void [auth_legacy_free](#) ([auth_legacy_t](#) *legacy)
- [auth_legacy_t](#) * [auth_legacy_alloc](#) (void)
- [auth_legacy_t](#) * [auth_legacy](#) ([stringer_t](#) *username, [stringer_t](#) *password)

Calculates the legacy symmetric encryption key and authentication token.

5.169.1 Function Documentation

5.169.1.1 [auth_legacy_t](#)* [auth_legacy](#) ([stringer_t](#) * username, [stringer_t](#) * password)

Calculates the legacy symmetric encryption key and authentication token. [legacy.c](#)

Parameters:

username a managed string holding the sanitized username.

password a managed string holding the plain text user password.

Returns:

an [auth_legacy_t](#) structure is returned upon success, and NULL upon failure.

Definition at line 60 of file [legacy.c](#).

References [auth_legacy_alloc\(\)](#), [auth_legacy_free\(\)](#), [CONTIGUOUS](#), [hash_sha512\(\)](#), [auth_legacy_t::key](#), [log_error](#), [magma](#), [MANAGED_T](#), [MANAGEDBUF](#), [magma_t::salt](#), [SECURE](#), [magma_t::secure](#), [st_alloc\(\)](#), [st_alloc_opts\(\)](#), [st_empty](#), [st_free\(\)](#), [st_merge](#), and [auth_legacy_t::token](#).

Referenced by [auth_login\(\)](#).

5.169.1.2 [auth_legacy_t](#)* [auth_legacy_alloc](#) (void)

Definition at line 43 of file [legacy.c](#).

References [mm_alloc\(\)](#), and [mm_wipe\(\)](#).

Referenced by [auth_legacy\(\)](#).

5.169.1.3 void [auth_legacy_free](#) ([auth_legacy_t](#) * legacy)

Definition at line 34 of file [legacy.c](#).

References [auth_legacy_t::key](#), [mm_free\(\)](#), [st_cleanup](#), and [auth_legacy_t::token](#).

Referenced by [auth_legacy\(\)](#), and [auth_login\(\)](#).

5.170 src/objects/auth/stacie.c File Reference

```
#include "magma.h"
```

Functions

- void [auth_stacie_free](#) ([auth_stacie_t](#) *stacie)
A checked cleanup function which can be used free a STACIE object.
- void [auth_stacie_cleanup](#) ([auth_stacie_t](#) *stacie)
A checked cleanup function which can be used free a STACIE object.
- [auth_stacie_t](#) * [auth_stacie_alloc](#) (void)
Allocate and initialize a STACIE object.
- [auth_stacie_t](#) * [auth_stacie](#) (uint32_t bonus, [stringer_t](#) *username, [stringer_t](#) *password, [stringer_t](#) *salt, [stringer_t](#) *verification, [stringer_t](#) *nonce)
Calculates the STACIE key and token values using the provided inputs.

5.170.1 Function Documentation

5.170.1.1 [auth_stacie_t](#) * [auth_stacie](#) (uint32_t *bonus*, [stringer_t](#) * *username*, [stringer_t](#) * *password*, [stringer_t](#) * *salt*, [stringer_t](#) * *verification*, [stringer_t](#) * *nonce*)

Calculates the STACIE key and token values using the provided inputs. [stacie.c](#)

Note:

If the password is NULL, then a nonce is required, and only the ephemeral token will be populated in the resulting [auth_stacie_t](#) structure. Likewise, if the nonce is NULL, then the password must be provided, and the ephemeral token will be absent in the resulting [auth_stacie_t](#) structure. If the password and nonce are both set to NULL, then an error is triggered and NULL is returned.

Parameters:

- bonus*** the number of bonus hash rounds to be applied for this particular user account.
- username*** a managed string holding the normalized username.
- password*** a managed string holding the plain text user password.
- salt*** a managed string with the salt value for the current user.
- nonce*** the randomly generated nonce value for the ephemeral token.

Returns:

- an [auth_stacie_t](#) structure is returned upon success, and NULL upon failure.

Definition at line 92 of file [stacie.c](#).

References [auth_stacie_alloc\(\)](#), [auth_stacie_free\(\)](#), [auth_stacie_t::ephemeral](#), [auth_stacie_t::keys](#), [log_pedantic](#), [auth_stacie_t::master](#), [auth_stacie_t::password](#), [st_empty](#), [stacie_derive_key\(\)](#), [stacie_derive_rounds\(\)](#), [stacie_derive_seed\(\)](#), [stacie_derive_token\(\)](#), [STACIE_KEY_ROUNDS_MAX](#), [auth_stacie_t::tokens](#), and [auth_stacie_t::verification](#).

Referenced by [auth_login\(\)](#), [auth_response\(\)](#), and [register_data_insert_user\(\)](#).

5.170.1.2 `auth_stacie_t* auth_stacie_alloc (void)`

Allocate and initialize a STACIE object.

Returns:

NULL on failure, or a pointer to the newly allocated STACIE object on success.

Definition at line 60 of file `stacie.c`.

References `log_pedantic`, `mm_alloc()`, and `mm_wipe()`.

Referenced by `auth_stacie()`.

5.170.1.3 `void auth_stacie_cleanup (auth_stacie_t * stacie)`

A checked cleanup function which can be used free a STACIE object.

Note:

Use this function when the STACIE object pointer could potentially be NULL. Call the free function directly if the STACIE object is guaranteed not to be NULL.

See also:

[auth_stacie_free\(\)](#)

Parameters:

stacie the STACIE object which will be freed.

Returns:

This function returns no value.

Definition at line 48 of file `stacie.c`.

References `auth_stacie_free()`.

Referenced by `auth_response()`, and `register_data_insert_user()`.

5.170.1.4 `void auth_stacie_free (auth_stacie_t * stacie)`

A checked cleanup function which can be used free a STACIE object.

Note:

Use this function when the STACIE object is guaranteed not to be NULL. Call the cleanup function directly if the STACIE object pointer could potentially be NULL.

See also:

[auth_stacie_cleanup\(\)](#)

Parameters:

stacie the STACIE object which will be freed.

Returns:

This function returns no value.

Definition at line 19 of file stacie.c.

References `auth_stacie_t::ephemeral`, `auth_stacie_t::keys`, `log_pedantic`, `auth_stacie_t::master`, `mm_free()`, `auth_stacie_t::password`, `st_cleanup`, `auth_stacie_t::tokens`, and `auth_stacie_t::verification`.

Referenced by `auth_login()`, `auth_response()`, `auth_stacie()`, and `auth_stacie_cleanup()`.

5.171 src/objects/auth/username.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * auth_sanitize_address \(stringer_t *username\)](#)
Process an email address or username to ensure it only contains valid characters.
- [stringer_t * auth_sanitize_username \(stringer_t *username\)](#)
Get the username portion of a fully qualified email address.

5.171.1 Function Documentation

5.171.1.1 stringer_t* auth_sanitize_address (stringer_t * username)

Process an email address or username to ensure it only contains valid characters. [username.c](#)

Parameters:

username a managed string containing the user supplied login name to be processed, it may contain an optional domain suffix.

Note:

This function duplicates the input address string, with all characters converted to lowercase, and whitespace removed. '.' and '-' are also converted to '_' in the username, and if there is a '+' in the local portion of an email address, all subsequent characters in that local part will be ignored, as they constitute a label.

Returns:

NULL on failure, or a managed string containing the sanitized address on success, that must be freed by the caller.

Definition at line 18 of file username.c.

References [chr_whitespace\(\)](#), [lower_chr\(\)](#), [PLACER](#), [st_alloc\(\)](#), [st_char_get\(\)](#), [st_empty_out\(\)](#), [st_length_get\(\)](#), and [st_length_set\(\)](#).

Referenced by [auth_sanitize_username\(\)](#), [smtp_data_outbound\(\)](#), and [smtp_rcpt_to\(\)](#).

5.171.1.2 stringer_t* auth_sanitize_username (stringer_t * username)

Get the username portion of a fully qualified email address.

Parameters:

username a managed string containing the username to be processed, with an optional domain suffix.

Returns:

NULL on failure, or a managed string containing the credential username portion of the supplied address.

Definition at line 103 of file username.c.

References [auth_sanitize_address\(\)](#), [magma_t::domain](#), [magma](#), [PLACER](#), [st_char_get\(\)](#), [st_cmp_cs_eq\(\)](#), [st_length_get\(\)](#), [st_length_set\(\)](#), [st_search_cs\(\)](#), and [magma_t::system](#).

Referenced by [api_endpoint_register\(\)](#), and [auth_challenge\(\)](#).

5.172 src/objects/config/config.c File Reference

```
#include "magma.h"
```

Functions

- void `user_config_entry_free` (`user_config_entry_t` *entry)
Destroy a user config entry.
- void `user_config_free` (`user_config_t` *collection)
Free a user config collection object.
- `user_config_t` * `user_config_alloc` (uint64_t usernum)
Allocate a user config collection object for a user.
- `user_config_entry_t` * `user_config_entry_alloc` (`stringer_t` *key, `stringer_t` *value, uint64_t flags)
Create and initialize new user config entry object.
- `user_config_t` * `user_config_create` (uint64_t usernum)
Create a new user config collection object for the specified user and populate it with config entries from the database.
- int_t `user_config_update` (`user_config_t` *collection)
Update a collection of user config options from the database, if necessary.
- int_t `user_config_edit` (`user_config_t` *collection, `stringer_t` *key, `stringer_t` *value)
Change the value of, or delete a key from a user's config collection.

5.172.1 Function Documentation

5.172.1.1 `user_config_t`* `user_config_alloc` (uint64_t usernum)

Allocate a user config collection object for a user. config.c

Parameters:

usernum the numerical user id for the requested user.

Returns:

NULL on failure or a pointer to the newly allocated user config collection object on success.

Definition at line 44 of file config.c.

References `align()`, `inx_alloc()`, `log_pedantic`, `M_INX_TREE`, `mm_alloc()`, `mm_free()`, `user_config_entry_free()`, and `user_config_t`.

Referenced by `user_config_create()`.

5.172.1.2 `user_config_t`* `user_config_create` (uint64_t usernum)

Create a new user config collection object for the specified user and populate it with config entries from the database.

Parameters:

usernum the numerical user id for the user to be queried.

Returns:

NULL on failure, or a pointer to the newly created user config collection object on success.

Definition at line 106 of file config.c.

References log_pedantic, OBJECT_CONFIG, serial_get(), user_config_alloc(), user_config_fetch(), user_config_free(), and user_config_t.

Referenced by portal_endpoint_config_edit(), and portal_endpoint_config_load().

5.172.1.3 int_t user_config_edit (user_config_t * collection, stringer_t * key, stringer_t * value)

Change the value of, or delete a key from a user's config collection.

Parameters:

collection a pointer to a collection of user's config entries.

key a pointer to a managed string containing the name of the user config key to be modified.

value a pointer to a managed string containing the name of the new value of the key, or NULL if it is to be deleted.

Returns:

-1 on error or 1 on success.

Definition at line 160 of file config.c.

References inx_delete(), inx_find(), inx_replace(), log_pedantic, M_TYPE_STRINGER, OBJECT_CONFIG, serial_increment(), multi_t::st, st_cmp_cs_eq(), st_empty, user_config_delete(), user_config_entry_alloc(), user_config_entry_free(), user_config_entry_t, user_config_upsert(), and multi_t::val.

Referenced by portal_endpoint_config_edit().

5.172.1.4 user_config_entry_t* user_config_entry_alloc (stringer_t * key, stringer_t * value, uint64_t flags)

Create and initialize new user config entry object.

Parameters:

key a managed string containing the key of the config entry object.

value a managed string containing the value of the config entry object.

flags a bitmask reflecting the flags of the config entry object.

Returns:

NULL on failure, or a pointer to the newly allocated and initialized user config entry object on success.

Definition at line 70 of file config.c.

References align(), FOREIGNDATA, JOINTED, log_pedantic, mm_alloc(), mm_copy(), PLACER_T, placer_t, st_data_get(), st_length_get(), STACK, and user_config_entry_t.

Referenced by user_config_edit(), and user_config_fetch().

5.172.1.5 void user_config_entry_free (user_config_entry_t * entry)

Destroy a user config entry.

Parameters:

entry a pointer to the user config entry object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file config.c.

References mm_free().

Referenced by user_config_alloc(), user_config_edit(), and user_config_fetch().

5.172.1.6 void user_config_free (user_config_t * collection)

Free a user config collection object.

Parameters:

collection a pointer to the user config collection object to be freed.

Returns:

This function returns no value.

Definition at line 29 of file config.c.

References inx_cleanup(), and mm_free().

Referenced by portal_endpoint_config_edit(), portal_endpoint_config_load(), and user_config_create().

5.172.1.7 int_t user_config_update (user_config_t * collection)

Update a collection of user config options from the database, if necessary.

Note:

This function is not currently being used anywhere. If this function returns -1, the config entries are left in an undefined state and should not be relied upon.

Parameters:

collection a pointer to a collection of user config entries to be checked for updates.

Returns:

1 if the collection is up-to-date, 0 if it was refreshed to update changes, or -1 if the refresh operation failed.

Definition at line 132 of file config.c.

References inx_truncate(), OBJECT_CONFIG, serial_get(), and user_config_fetch().

5.173 src/web/portal/config.c File Reference

```
#include "magma.h"
```

Functions

- `json_t * portal_config_entry_flags (user_config_entry_t *entry)`
Get a user config entry's flags as a json object.
- `json_t * portal_config_entry (user_config_entry_t *entry)`
- `json_t * portal_config_collection (user_config_t *collection)`
Get a collection of user config options as a json object.

5.173.1 Function Documentation

5.173.1.1 `json_t* portal_config_collection (user_config_t *collection)`

Get a collection of user config options as a json object. config.c

Parameters:

collection a pointer to a user config object that contains all the config options pairs to be serialized.

Returns:

NULL on failure, or a pointer to a json object containing the collection of user config options on success.

Definition at line 62 of file config.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `json_decref_d`, `json_object_d`, `json_object_set_new_d`, `log_error`, `log_options`, `M_LOG_CRITICAL`, `portal_config_entry()`, `st_char_get()`, `USER_CONF_STATUS_CRITICAL`, and `user_config_entry_t`.

Referenced by `portal_endpoint_config_load()`.

5.173.1.2 `json_t* portal_config_entry (user_config_entry_t *entry)`

Definition at line 35 of file config.c.

References `json_decref_d`, `json_pack_ex_d`, `log_pedantic`, `portal_config_entry_flags()`, `st_char_get()`, and `st_length_int()`.

Referenced by `portal_config_collection()`.

5.173.1.3 `json_t* portal_config_entry_flags (user_config_entry_t *entry)`

Get a user config entry's flags as a json object.

Parameters:

entry a pointer to the user config entry object to have its flags serialized.

Returns:

NULL on failure or a pointer to a json object storing the user's flags on success.

Definition at line 15 of file config.c.

References `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_string_d`, and `USER_CONF_STATUS_CRITICAL`.

Referenced by `portal_config_entry()`.

5.174 src/objects/contacts/contacts.c File Reference

```
#include "magma.h"
```

Functions

- void [contact_free](#) ([contact_t](#) *contact)
Free a contact entry object.
- void [contact_detail_free](#) ([contact_detail_t](#) *detail)
Free a contact entry detail.
- [contact_t](#) * [contact_alloc](#) (uint64_t contactnum, [stringer_t](#) *name)
Create a new contact entry object.
- [contact_detail_t](#) * [contact_detail_alloc](#) ([stringer_t](#) *key, [stringer_t](#) *value, uint64_t flags)
Create a new contact detail from a key-value pair.
- void [contact_name](#) ([contact_t](#) *contact, [stringer_t](#) *name)
Update the name (in memory) of a contact entry.
- [inx_t](#) * [contacts_update](#) (uint64_t usernum)
Retrieve all of a user's contacts (and contact folders) from the database.
- [contact_t](#) * [contact_create](#) (uint64_t usernum, uint64_t foldernum, [stringer_t](#) *name)
Create a new contact entry object and insert it into the database.
- [int_t](#) [contact_remove](#) ([contact_t](#) *contact, uint64_t usernum, uint64_t foldernum)
Remove a contact entry and its associated details from the database.
- [int_t](#) [contact_move](#) ([contact_folder_t](#) *source, [contact_folder_t](#) *target, [contact_t](#) *contact, uint64_t usernum)
Move a contact entry from one contact to another.
- [int_t](#) [contact_edit](#) ([contact_t](#) *contact, uint64_t usernum, uint64_t foldernum, [stringer_t](#) *key, [stringer_t](#) *value)
Change a detail of a contact entry.
- [bool_t](#) [contact_validate_name](#) ([stringer_t](#) *name, [chr_t](#) **errmsg)
Validate the name of a requested contact entry name.
- [bool_t](#) [contact_validate_detail](#) ([stringer_t](#) *key, [stringer_t](#) *value, [chr_t](#) **errmsg)
Validate the value of a requested contact entry detail.

5.174.1 Function Documentation

5.174.1.1 [contact_t](#)* [contact_alloc](#) (uint64_t contactnum, [stringer_t](#) * name)

Create a new contact entry object. contacts.c

Parameters:

contactnum the numerical id of the new contact entry.

name a managed string containing the name of the contact.

Returns:

NULL on failure, or a pointer to the newly allocated and initialized contact entry object on success.

Definition at line 48 of file contacts.c.

References align(), contact_detail_free(), contact_t, FOREIGNDATA, inx_alloc(), JOINTED, log_pedantic, M_INX_TREE, mm_alloc(), mm_copy(), mm_free(), PLACER_T, placer_t, st_data_get(), st_length_get(), and STACK.

Referenced by contact_create(), contact_move(), and contacts_fetch().

5.174.1.2 contact_t* contact_create (uint64_t usernum, uint64_t foldernum, stringer_t * name)

Create a new contact entry object and insert it into the database.

Parameters:

usernum the numerical id of the user to whom the new contact will belong.

foldernum the numerical id of the parent folder that will contain the contact entry.

name a pointer to a managed string containing the name of the new contact entry.

Returns:

NULL on failure, or a pointer to the newly created contact entry object on success.

Definition at line 185 of file contacts.c.

References contact_alloc(), contact_insert(), contact_t, log_pedantic, and st_empty.

Referenced by portal_endpoint_contacts_add(), and portal_endpoint_contacts_copy().

5.174.1.3 contact_detail_t* contact_detail_alloc (stringer_t * key, stringer_t * value, uint64_t flags)

Create a new contact detail from a key-value pair.

Parameters:

key a managed string containing the name of the detail.

value a managed string containing the value associated with the detail.

flags a bitmask of flags associated with the detail.

Returns:

NULL on failure, or a pointer to the newly allocated and initialized contact detail object on success.

Definition at line 79 of file contacts.c.

References align(), contact_detail_t, FOREIGNDATA, JOINTED, log_pedantic, mm_alloc(), mm_copy(), PLACER_T, placer_t, st_data_get(), st_length_get(), and STACK.

Referenced by contact_details_fetch(), and contact_edit().

5.174.1.4 void contact_detail_free (contact_detail_t * detail)

Free a contact entry detail.

Parameters:

detail a pointer to the contact entry detail to be freed.

Returns:

This function returns no value.

Definition at line 36 of file contacts.c.

References mm_cleanup.

Referenced by contact_alloc(), contact_details_fetch(), and contact_edit().

5.174.1.5 int_t contact_edit (contact_t * contact, uint64_t usernum, uint64_t foldernum, stringer_t * key, stringer_t * value)

Change a detail of a contact entry.

Parameters:

contact a pointer to the target contact entry.

usernum the numerical id of the user to whom the specified contact entry belongs.

foldernum

key a managed string containing the name of the contact entry detail to be changed. If the special value "name" is passed as the key, the contact name will be changed instead.

value a managed string containing the new value of the specified contact entry detail; if NULL is passed, the contact detail will be deleted rather than modified.

Returns:

-1 on failure or 1 on success.

LOW: It would be nice if we didn't call this function each time we updated a contact, but rather we called it when all of the updates have been completed.

LOW: It would be nice if we didn't call this function each time we updated a contact, but rather we called it when all of the updates have been completed.

Definition at line 274 of file contacts.c.

References contact_detail_alloc(), contact_detail_delete(), contact_detail_free(), contact_detail_t, contact_detail_upsert(), contact_name(), contact_update(), contact_update_stamp(), inx_delete(), inx_replace(), log_pedantic, M_TYPE_STRINGER, PLACER, multi_t::st, st_cmp_ci_eq(), st_empty, and multi_t::val.

Referenced by portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), and portal_endpoint_contacts_edit().

5.174.1.6 void contact_free (contact_t * contact)

Free a contact entry object.

Parameters:

contact a pointer to the contact entry to be freed.

Returns:

This function returns no value.

Definition at line 16 of file contacts.c.

References FOREIGNDATA, inx_cleanup(), mm_free(), st_free(), and st_opt_test().

Referenced by contact_folder_alloc(), contact_move(), contacts_fetch(), portal_endpoint_contacts_add(), and portal_endpoint_contacts_copy().

5.174.1.7 `int_t contact_move (contact_folder_t * source, contact_folder_t * target, contact_t * contact, uint64_t usernum)`

Move a contact entry from one contact to another.

Parameters:

source a pointer to the original parent contact folder object of the contact entry to be moved.

target a pointer to the contact folder object that will become the new parent of the specified contact entry.

contact a pointer to the contact object that is to be moved.

usernum the numerical id of the user to whom the specified contact entry belongs.

Returns:

-1 on error, or 1 if the contact move operation was successful.

Definition at line 229 of file contacts.c.

References `contact_alloc()`, `contact_free()`, `contact_t`, `contact_update()`, `inx_delete()`, `inx_insert()`, `inx_t`, `log_pedantic`, `M_TYPE_UINT64`, `OBJECT_CONTACTS`, `serial_increment()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_endpoint_contacts_move()`.

5.174.1.8 `void contact_name (contact_t * contact, stringer_t * name)`

Update the name (in memory) of a contact entry.

Parameters:

contact a pointer to the contact entry object to be updated.

name a managed string containing the new name of the specified contact entry.

Returns:

This function returns no value.

HIGH: A possible segfault is exposed here, related to the `contacts.edit camelface` call and having to do with the way foreign data is handled.

Definition at line 118 of file contacts.c.

References `data`, `FOREIGNDATA`, `JOINTED`, `mm_dupe()`, `mm_free()`, `st_data_get()`, `st_data_set()`, `st_empty_out()`, `st_length_set()`, `st_opt_set()`, and `st_opt_test()`.

Referenced by `contact_edit()`.

5.174.1.9 `int_t contact_remove (contact_t * contact, uint64_t usernum, uint64_t foldernum)`

Remove a contact entry and its associated details from the database.

See also:

[contact_delete\(\)](#)

Note:

This function is not currently in use anywhere in the code.

Parameters:

contact a pointer to the contact entry to be deleted from the database.

usernum the numerical id of the user to whom the contact entry belongs.

foldernum the numerical id of the parent folder containing the specified contact entry.

Returns:

1 if the specified contact was deleted successfully, 0 if no matching contact was found in the database, or -1 on general failure.

Definition at line 211 of file contacts.c.

References contact_delete(), and log_pedantic.

5.174.1.10 bool_t contact_validate_detail (stringer_t * *key*, stringer_t * *value*, chr_t ** *errmsg*)

Validate the value of a requested contact entry detail.

Parameters:

key a pointer to a managed string containing the name of the contact detail key to be validated.

value a pointer to a managed string containing the value of the contact detail to be validated.

errmsg if not NULL, the address of a string pointer that will receive a descriptive error message upon validation failure.

Returns:

true if the proposed contact key/value pair was valid, or false if it was not.

Definition at line 373 of file contacts.c.

References chr_printable(), and st_empty_out().

Referenced by portal_endpoint_contacts_add().

5.174.1.11 bool_t contact_validate_name (stringer_t * *name*, chr_t ** *errmsg*)

Validate the name of a requested contact entry name.

Parameters:

name a pointer to a managed string containing the name of the contact to be validated.

errmsg if not NULL, the address of a string pointer that will receive a descriptive error message upon validation failure.

Returns:

true if the proposed contact name was valid, or false if it was not.

Definition at line 338 of file contacts.c.

References chr_printable(), and st_empty_out().

Referenced by portal_endpoint_contacts_add().

5.174.1.12 inx_t* contacts_update (uint64_t *usernum*)

Retrieve all of a user's contacts (and contact folders) from the database.

Parameters:

usernum the numerical id of the user whose contact folders and contact entries should be fetched.

Returns:

NULL on failure, or a pointer to an `inx` holder containing all of the specified user's contact folders and entries on success.

Definition at line 152 of file `contacts.c`.

References `contacts_fetch()`, `inx_cleanup()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `log_pedantic`, `M_FOLDER_CONTACTS`, and `magma_folder_fetch()`.

Referenced by `meta_update_contacts()`.

5.175 src/objects/folders/contacts.c File Reference

```
#include "magma.h"
```

Functions

- void [contact_folder_free](#) ([contact_folder_t](#) *folder)
Free a contact folder object.
- [contact_folder_t](#) * [contact_folder_alloc](#) (uint64_t foldernum, uint64_t parent, uint32_t order, [stringer_t](#) *name)
Allocate and initialize a new contact folder object.
- [int_t](#) [contact_folder_create](#) (uint64_t usernum, uint64_t parent, [stringer_t](#) *name, [inx_t](#) *folders)
Create a new contact folder.
- [int_t](#) [contact_folder_remove](#) (uint64_t usernum, uint64_t foldernum, [inx_t](#) *folders)
Remove a user's contact folder.
- [bool_t](#) [contact_folder_rename](#) (uint64_t usernum, uint64_t foldernum, [stringer_t](#) *rename, [inx_t](#) *folders)
Rename a user's contact folder.

5.175.1 Function Documentation

5.175.1.1 [contact_folder_t](#)* [contact_folder_alloc](#) (uint64_t foldernum, uint64_t parent, uint32_t order, [stringer_t](#) * name)

Allocate and initialize a new contact folder object. contacts.c

Parameters:

- foldernum*** the numerical id of this contact folder.
- parent*** the numerical id of the parent folder of this contact folder.
- order*** the order number of this folder in its parent folder.
- name*** a managed string containing the name of this folder.

Returns:

NULL on failure, or a pointer to the newly initialized contact folder object on success.

Definition at line 29 of file contacts.c.

References [contact_free\(\)](#), [inx_alloc\(\)](#), [M_INX_TREE](#), [magma_folder_alloc\(\)](#), and [mm_cleanup](#).

Referenced by [contact_folder_create\(\)](#), and [magma_folder_funcs\(\)](#).

5.175.1.2 [int_t](#) [contact_folder_create](#) (uint64_t usernum, uint64_t parent, [stringer_t](#) * name, [inx_t](#) * folders)

Create a new contact folder.

Parameters:

- usernum*** the numerical id of the user to whom the new imap folder will belong.
- parent*** the numerical id of the parent folder to contain the new contact folder, or 0 for a root folder.
- name*** a managed string containing the fully qualified name of the new contact folder to be created.

folders an `inx` holder containing the set of folders into which the new contact folder will be inserted.

Returns:

1 on success or ≤ 0 on failure. 0: An invalid parameter was passed to the function, or an internal failure occurred. -1: The specified new folder name was invalid. -2: Part of the folder path name exceeds 16 characters (FOLDER_LENGTH_LIMIT) after being unescaped for quotation marks. -3: The specified folder name already exists. -4: An invalid parent folder was specified. -5: Part of the folder path name exceeds 16 characters (FOLDER_LENGTH_LIMIT) after being unescaped for quotation marks.

Definition at line 55 of file `contacts.c`.

References `contact_folder_alloc()`, `contact_folder_free()`, `FOLDER_LENGTH_LIMIT`, `IMAP_FOLDER_RECURSION_LMIIT`, `imap_valid_folder_name()`, `inx_insert()`, `log_pedantic`, `M_FOLDER_CONTACTS`, `M_TYPE_UINT64`, `magma_folder_children()`, `magma_folder_find_name()`, `magma_folder_find_number()`, `magma_folder_insert()`, `magma_folder_t`, `st_empty`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_folder_contacts_add()`.

5.175.1.3 void contact_folder_free (contact_folder_t *folder)

Free a contact folder object.

Parameters:

folder a pointer to the contact folder object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file `contacts.c`.

References `magma_folder_free()`.

Referenced by `contact_folder_create()`, and `magma_folder_funcs()`.

5.175.1.4 int_t contact_folder_remove (uint64_t usernum, uint64_t foldernum, inx_t *folders)

Remove a user's contact folder.

See also:

[magma_folder_delete\(\)](#)

Note:

This operation will only succeed if the specified contact folder is empty.

Parameters:

usernum the numerical id of the user requesting the contact folder removal.

foldernum the numerical id of the folder to be removed.

folders an `inx` holder containing the contact folders to be searched for the folder specified for removal.

Returns:

0 on success or < 0 on failure. -1: There was an unexpected internal error. -2: The specified folder could not be found. -3: Contact folder was not empty.

Definition at line 121 of file contacts.c.

References `inx_count()`, `inx_delete()`, `inx_find()`, `log_pedantic`, `M_FOLDER_CONTACTS`, `M_TYPE_UINT64`, `magma_folder_children()`, and `magma_folder_delete()`.

Referenced by `portal_folder_contacts_remove()`.

5.175.1.5 `bool_t contact_folder_rename (uint64_t usernum, uint64_t foldernum, stringer_t * rename, inx_t * folders)`

Rename a user's contact folder.

See also:

[magma_folder_rename\(\)](#)

Parameters:

usernum the numerical id of the user requesting the contact folder renaming.

foldernum the numerical id of the folder to be renamed.

rename a managed string containing the new name of the specified folder.

folders an inx holder containing the contact folders to be searched for the folder specified to be renamed.

Returns:

true on success or false on failure.

Definition at line 161 of file contacts.c.

References `FOLDER_LENGTH_LIMIT`, `imap_valid_folder_name()`, `inx_find()`, `log_pedantic`, `M_FOLDER_CONTACTS`, `M_TYPE_UINT64`, `magma_folder_find_name()`, `magma_folder_rename()`, `st_dupe()`, `st_empty`, `st_free()`, and `st_length_get()`.

Referenced by `portal_endpoint_folders_rename()`.

5.176 src/web/portal/contacts.c File Reference

```
#include "magma.h"
```

Functions

- `json_t * portal_contact_detail_flags (contact_detail_t *detail)`

TODO: Right now this function doesn't really do anything. It needs to be updated if/once flag values are determined.

- `json_t * portal_contact_details (contact_t *contact)`

Retrieve all the details about a contact entry as a json object.

5.176.1 Function Documentation

5.176.1.1 `json_t* portal_contact_detail_flags (contact_detail_t * detail)`

TODO: Right now this function doesn't really do anything. It needs to be updated if/once flag values are determined. `contacts.c`

Pack a json array representing the flags associated with a user contact entry detail.

Parameters:

detail a pointer to the user contact detail to have its flags queried.

Returns:

NULL on failure or a pointer to a json array containing strings describing the specified contact detail's flags.

Definition at line 16 of file `contacts.c`.

References `CONTACT_DETAIL_FLAG_CRITICAL`, `json_array_append_new_d`, `json_array_d`, and `json_string_d`.

Referenced by `portal_contact_details()`.

5.176.1.2 `json_t* portal_contact_details (contact_t * contact)`

Retrieve all the details about a contact entry as a json object.

Parameters:

contact a pointer to the contact object whose details will be retrieved.

Returns:

a pointer to a json object containing the details of the specified contact entry.

Definition at line 35 of file `contacts.c`.

References `contact_detail_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `json_array_d`, `json_decref_d`, `json_object_set_new_d`, `json_pack_ex_d`, `log_error`, `log_pedantic`, `portal_contact_detail_flags()`, and `st_char_get()`.

Referenced by `portal_endpoint_contacts_load()`.

5.177 src/objects/contacts/contacts.h File Reference

Enumerations

- enum { [CONTACT_DETAIL_FLAG_NONE](#) = 0, [CONTACT_DETAIL_FLAG_CRITICAL](#) = 1 }

Functions

- struct [__attribute__](#) ((packed))
[contact_t](#) * [contact_alloc](#) (uint64_t contactnum, [stringer_t](#) *name)
contacts.c
- [contact_t](#) * [contact_create](#) (uint64_t usernum, uint64_t foldernum, [stringer_t](#) *name)
Create a new contact entry object and insert it into the database.
- [contact_detail_t](#) * [contact_detail_alloc](#) ([stringer_t](#) *key, [stringer_t](#) *value, uint64_t flags)
Create a new contact detail from a key-value pair.
- void [contact_detail_free](#) ([contact_detail_t](#) *detail)
Free a contact entry detail.
- int_t [contact_edit](#) ([contact_t](#) *contact, uint64_t usernum, uint64_t foldernum, [stringer_t](#) *key, [stringer_t](#) *value)
Change a detail of a contact entry.
- void [contact_free](#) ([contact_t](#) *contact)
Free a contact entry object.
- int_t [contact_move](#) ([contact_folder_t](#) *source, [contact_folder_t](#) *target, [contact_t](#) *contact, uint64_t usernum)
Move a contact entry from one contact to another.
- void [contact_name](#) ([contact_t](#) *contact, [stringer_t](#) *name)
Update the name (in memory) of a contact entry.
- int_t [contact_remove](#) ([contact_t](#) *contact, uint64_t usernum, uint64_t foldernum)
Remove a contact entry and its associated details from the database.
- bool_t [contact_validate_name](#) ([stringer_t](#) *name, [chr_t](#) **errmsg)
Validate the name of a requested contact entry name.
- bool_t [contact_validate_detail](#) ([stringer_t](#) *key, [stringer_t](#) *value, [chr_t](#) **errmsg)
Validate the value of a requested contact entry detail.
- inx_t * [contacts_update](#) (uint64_t usernum)
Retrieve all of a user's contacts (and contact folders) from the database.
- int_t [contact_delete](#) (uint64_t contactnum, uint64_t usernum, uint64_t foldernum)
datatier.c
- int_t [contact_detail_delete](#) (uint64_t contactnum, [stringer_t](#) *key)
Delete the specified contact detail of a contact entry from the database.
- int_t [contact_detail_upsert](#) (uint64_t contactnum, [stringer_t](#) *key, [stringer_t](#) *value, uint64_t flags)

Update a specified contact detail in the database, or insert it if it does not exist.

- [int_t contact_details_fetch](#) ([contact_t](#) *contact)

Populate a contact entry with its details from the database.

- [uint64_t contact_insert](#) ([uint64_t](#) usernum, [uint64_t](#) foldernum, [stringer_t](#) *name)

Insert a new contact entry into the database.

- [int_t contact_update](#) ([uint64_t](#) contactnum, [uint64_t](#) usernum, [uint64_t](#) cur_folder, [uint64_t](#) target_folder, [stringer_t](#) *name)

Update a user contact entry in the database.

- [int_t contact_update_stamp](#) ([uint64_t](#) contactnum, [uint64_t](#) usernum, [uint64_t](#) foldernum)

Touch the last updated time stamp of a contact entry in the database.

- [int_t contacts_fetch](#) ([uint64_t](#) usernum, [contact_folder_t](#) *folder)

Retrieve all of a user's contact entries in a specified contacts folder from the database.

- [contact_t](#) * [contact_find_detail](#) ([contact_folder_t](#) *folder, [stringer_t](#) *key, [stringer_t](#) *target)

find.c

- [contact_t](#) * [contact_find_name](#) ([contact_folder_t](#) *folder, [stringer_t](#) *target)

Get a contact entry by name (case insensitive).

- [contact_t](#) * [contact_find_number](#) ([contact_folder_t](#) *folder, [uint64_t](#) target)

Find a contact entry in a contact folder by number.

Variables

- [contact_detail_t](#)
- [contact_t](#)

5.177.1 Enumeration Type Documentation

5.177.1.1 anonymous enum

Enumerator:

CONTACT_DETAIL_FLAG_NONE

CONTACT_DETAIL_FLAG_CRITICAL

Definition at line 11 of file contacts.h.

5.177.2 Function Documentation

5.177.2.1 struct __attribute__((packed)) [read]

Definition at line 21 of file contacts.h.

References [inx_t](#).

5.177.2.2 contact_t* contact_alloc (uint64_t *contactnum*, stringer_t * *name*)

contacts.c contacts.c

Parameters:

contactnum the numerical id of the new contact entry.
name a managed string containing the name of the contact.

Returns:

NULL on failure, or a pointer to the newly allocated and initialized contact entry object on success.

Definition at line 48 of file contacts.c.

References align(), contact_detail_free(), contact_t, FOREIGNDATA, inx_alloc(), JOINTED, log_pedantic, M_INX_TREE, mm_alloc(), mm_copy(), mm_free(), PLACER_T, placer_t, st_data_get(), st_length_get(), and STACK.

Referenced by contact_create(), contact_move(), and contacts_fetch().

5.177.2.3 contact_t* contact_create (uint64_t *usernum*, uint64_t *foldernum*, stringer_t * *name*)

Create a new contact entry object and insert it into the database.

Parameters:

usernum the numerical id of the user to whom the new contact will belong.
foldernum the numerical id of the parent folder that will contain the contact entry.
name a pointer to a managed string containing the name of the new contact entry.

Returns:

NULL on failure, or a pointer to the newly created contact entry object on success.

Definition at line 185 of file contacts.c.

References contact_alloc(), contact_insert(), contact_t, log_pedantic, and st_empty.

Referenced by portal_endpoint_contacts_add(), and portal_endpoint_contacts_copy().

5.177.2.4 int_t contact_delete (uint64_t *contactnum*, uint64_t *usernum*, uint64_t *foldernum*)

datatier.c datatier.c

Parameters:

contactnum the numerical id of the contact entry to be deleted.
usernum the numerical id of the user to whom the specified contact entry belongs.
foldernum the numerical id of the parent contact folder containing the contact entry.

Returns:

1 if the specified contact was deleted successfully, 0 if no matching contact was found in the database, or -1 on general failure.

Definition at line 61 of file datatier.c.

References log_check, log_pedantic, mm_wipe(), and stmt_exec_affected().

Referenced by contact_remove(), and portal_endpoint_contacts_remove().

5.177.2.5 **contact_detail_t* contact_detail_alloc (stringer_t * *key*, stringer_t * *value*, uint64_t *flags*)**

Create a new contact detail from a key-value pair.

Parameters:

key a managed string containing the name of the detail.

value a managed string containing the value associated with the detail.

flags a bitmask of flags associated with the detail.

Returns:

NULL on failure, or a pointer to the newly allocated and initialized contact detail object on success.

Definition at line 79 of file contacts.c.

References align(), contact_detail_t, FOREIGNDATA, JOINTED, log_pedantic, mm_alloc(), mm_copy(), PLACER_T, placer_t, st_data_get(), st_length_get(), and STACK.

Referenced by contact_details_fetch(), and contact_edit().

5.177.2.6 **int_t contact_detail_delete (uint64_t *contactnum*, stringer_t * *key*)**

Delete the specified contact detail of a contact entry from the database.

Parameters:

contactnum the numerical id of the contact entry to have the specified detail removed.

key a managed string containing the name of the contact detail to be removed from the entry.

Returns:

-1 on error, 0 if no matching detail was found in the database, or 1 if the delete operation was successful.

Definition at line 204 of file datatier.c.

References log_check, log_pedantic, mm_wipe(), st_char_get(), st_length_get(), st_length_int(), and stmt_exec_affected().

Referenced by contact_edit().

5.177.2.7 **void contact_detail_free (contact_detail_t * *detail*)**

Free a contact entry detail.

Parameters:

detail a pointer to the contact entry detail to be freed.

Returns:

This function returns no value.

Definition at line 36 of file contacts.c.

References mm_cleanup.

Referenced by contact_alloc(), contact_details_fetch(), and contact_edit().

5.177.2.8 `int_t contact_detail_upsert (uint64_t contactnum, stringer_t * key, stringer_t * value, uint64_t flags)`

Update a specified contact detail in the database, or insert it if it does not exist.

Parameters:

- contactnum* the numerical id of the contact entry to be modified.
- key* a managed string containing the name of the contact detail to be updated.
- value* a managed string containing the new value of the specified contact detail.
- flags* a bitmask of flags to be associated with the specified contact entry detail.

Returns:

-1 on error, 0 if no update was necessary, 1 if a new contact detail was inserted into the database, or 2 if the specified contact detail was updated successfully.

Definition at line 241 of file `datatier.c`.

References `log_check`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `st_length_int()`, and `stmt_exec_affected()`.

Referenced by `contact_edit()`.

5.177.2.9 `int_t contact_details_fetch (contact_t * contact)`

Populate a contact entry with its details from the database.

Parameters:

- contact* a pointer to the contact entry object that will be updated.

Returns:

-1 on failure or 1 on success.

Definition at line 285 of file `datatier.c`.

References `contact_detail_alloc()`, `contact_detail_free()`, `contact_detail_t`, `inx_insert()`, `log_info`, `log_pedantic`, `M_TYPE_STRINGER`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `multi_t::st`, `stmt_get_result()`, and `multi_t::val`.

Referenced by `contacts_fetch()`.

5.177.2.10 `int_t contact_edit (contact_t * contact, uint64_t userid, uint64_t folderid, stringer_t * key, stringer_t * value)`

Change a detail of a contact entry.

Parameters:

- contact* a pointer to the target contact entry.
- userid* the numerical id of the user to whom the specified contact entry belongs.
- folderid*
- key* a managed string containing the name of the contact entry detail to be changed. If the special value "name" is passed as the key, the contact name will be changed instead.
- value* a managed string containing the new value of the specified contact entry detail; if NULL is passed, the contact detail will be deleted rather than modified.

Returns:

-1 on failure or 1 on success.

LOW: It would be nice if we didn't call this function each time we updated a contact, but rather we called it when all of the updates have been completed.

LOW: It would be nice if we didn't call this function each time we updated a contact, but rather we called it when all of the updates have been completed.

Definition at line 274 of file contacts.c.

References `contact_detail_alloc()`, `contact_detail_delete()`, `contact_detail_free()`, `contact_detail_t`, `contact_detail_upsert()`, `contact_name()`, `contact_update()`, `contact_update_stamp()`, `inx_delete()`, `inx_replace()`, `log_pedantic`, `M_TYPE_STRINGER`, `PLACER`, `multi_t::st`, `st_cmp_ci_eq()`, `st_empty`, and `multi_t::val`.

Referenced by `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, and `portal_endpoint_contacts_edit()`.

5.177.2.11 `contact_t* contact_find_detail (contact_folder_t * folder, stringer_t * key, stringer_t * target)`

find.c

Note:

This is a bit of a weird function that currently isn't being called from anywhere. It may or may not be here to stay.

Definition at line 13 of file find.c.

References `contact_detail_t`, `contact_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_find()`, `M_TYPE_STRINGER`, `st_cmp_ci_eq()`, and `st_empty`.

5.177.2.12 `contact_t* contact_find_name (contact_folder_t * folder, stringer_t * target)`

Get a contact entry by name (case insensitive).

Parameters:

folder a pointer to the contact folder to have its contents searched for the entry.

target a managed string containing the name of the target entry.

Returns:

NULL on failure or a pointer to the found user contact entry on success.

Definition at line 50 of file find.c.

References `contact_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `st_cmp_ci_eq()`, and `st_empty`.

5.177.2.13 `contact_t* contact_find_number (contact_folder_t * folder, uint64_t target)`

Find a contact entry in a contact folder by number.

Parameters:

folder a pointer to a contact folder object containing the contact entries to be searched.

target the numerical id of the contact entry to be located in the specified folder.

Returns:

NULL on failure or a pointer to the found user contact entry on success.

Definition at line 79 of file find.c.

References `contact_t`, `inx_find()`, `log_pedantic`, and `M_TYPE_UINT64`.

Referenced by `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_move()`, and `portal_endpoint_contacts_remove()`.

5.177.2.14 void contact_free (contact_t * *contact*)

Free a contact entry object.

Parameters:

contact a pointer to the contact entry to be freed.

Returns:

This function returns no value.

Definition at line 16 of file contacts.c.

References FOREIGNDATA, inx_cleanup(), mm_free(), st_free(), and st_opt_test().

Referenced by contact_folder_alloc(), contact_move(), contacts_fetch(), portal_endpoint_contacts_add(), and portal_endpoint_contacts_copy().

5.177.2.15 uint64_t contact_insert (uint64_t *usernum*, uint64_t *foldernum*, stringer_t * *name*)

Insert a new contact entry into the database.

Parameters:

usernum the numerical id of the user to whom the contact entry belongs.

foldernum the numerical id of the parent folder to contain the contact entry.

name a pointer to a managed string containing the name of the new contact entry.

Returns:

-1 on failure, 0 if no item was inserted into the database, or the id of the newly inserted contact entry in the database on success.

Definition at line 172 of file datatier.c.

References mm_wipe(), st_char_get(), st_length_get(), and stmt_insert().

Referenced by contact_create().

5.177.2.16 int_t contact_move (contact_folder_t * *source*, contact_folder_t * *target*, contact_t * *contact*, uint64_t *usernum*)

Move a contact entry from one contact to another.

Parameters:

source a pointer to the original parent contact folder object of the contact entry to be moved.

target a pointer to the contact folder object that will become the new parent of the specified contact entry.

contact a pointer to the contact object that is to be moved.

usernum the numerical id of the user to whom the specified contact entry belongs.

Returns:

-1 on error, or 1 if the contact move operation was successful.

Definition at line 229 of file contacts.c.

References contact_alloc(), contact_free(), contact_t, contact_update(), inx_delete(), inx_insert(), inx_t, log_pedantic, M_TYPE_UINT64, OBJECT_CONTACTS, serial_increment(), multi_t::u64, and multi_t::val.

Referenced by portal_endpoint_contacts_move().

5.177.2.17 void contact_name (contact_t * contact, stringer_t * name)

Update the name (in memory) of a contact entry.

Parameters:

contact a pointer to the contact entry object to be updated.

name a managed string containing the new name of the specified contact entry.

Returns:

This function returns no value.

HIGH: A possible segfault is exposed here, related to the contacts.edit camelface call and having to do with the way foreign data is handled.

Definition at line 118 of file contacts.c.

References data, FOREIGNDATA, JOINTED, mm_dupe(), mm_free(), st_data_get(), st_data_set(), st_empty_out(), st_length_set(), st_opt_set(), and st_opt_test().

Referenced by contact_edit().

5.177.2.18 int_t contact_remove (contact_t * contact, uint64_t usernum, uint64_t foldernum)

Remove a contact entry and its associated details from the database.

See also:

[contact_delete\(\)](#)

Note:

This function is not currently in use anywhere in the code.

Parameters:

contact a pointer to the contact entry to be deleted from the database.

usernum the numerical id of the user to whom the contact entry belongs.

foldernum the numerical id of the parent folder containing the specified contact entry.

Returns:

1 if the specified contact was deleted successfully, 0 if no matching contact was found in the database, or -1 on general failure.

Definition at line 211 of file contacts.c.

References contact_delete(), and log_pedantic.

5.177.2.19 int_t contact_update (uint64_t contactnum, uint64_t usernum, uint64_t cur_folder, uint64_t target_folder, stringer_t * name)

Update a user contact entry in the database.

Parameters:

contactnum the numerical id of the contact entry to be modified.

usernum the numerical id of the user to whom the specified contact entry belongs.

cur_folder the numerical id of the parent contact containing the specified contact entry.

target_folder if not 0, sets the new parent contact folder to which the specified contact entry will belong.

name if not NULL, sets the new name of the specified contact entry.

Returns:

-1 on error, 0 if the specified contact entry was not found in the database, or 1 if the contact entry was successfully updated.

Definition at line 106 of file `datatier.c`.

References `ISNULL`, `log_check`, `log_pedantic`, `mm_wipe()`, `st_char_get()`, `st_empty`, `st_length_get()`, and `stmt_exec_affected()`.

Referenced by `contact_edit()`, and `contact_move()`.

5.177.2.20 `int_t contact_update_stamp (uint64_t contactnum, uint64_t usernum, uint64_t foldernum)`

Touch the last updated time stamp of a contact entry in the database.

Parameters:

contactnum the numerical id of the contact entry to have its time stamp updated.

usernum the numerical id of the user to whom the contact entry belongs.

foldernum the numerical id of the parent folder containing the contact entry.

Returns:

1 if the contact time stamp was updated successfully, 0 if the specified contact was not found, or -1 on general failure.

Definition at line 17 of file `datatier.c`.

References `log_check`, `log_pedantic`, `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `contact_edit()`.

5.177.2.21 `bool_t contact_validate_detail (stringer_t * key, stringer_t * value, chr_t ** errmsg)`

Validate the value of a requested contact entry detail.

Parameters:

key a pointer to a managed string containing the name of the contact detail key to be validated.

value a pointer to a managed string containing the value of the contact detail to be validated.

errmsg if not NULL, the address of a string pointer that will receive a descriptive error message upon validation failure.

Returns:

true if the proposed contact key/value pair was valid, or false if it was not.

Definition at line 373 of file `contacts.c`.

References `chr_printable()`, and `st_empty_out()`.

Referenced by `portal_endpoint_contacts_add()`.

5.177.2.22 `bool_t contact_validate_name (stringer_t * name, chr_t ** errmsg)`

Validate the name of a requested contact entry name.

Parameters:

name a pointer to a managed string containing the name of the contact to be validated.

errmsg if not NULL, the address of a string pointer that will receive a descriptive error message upon validation failure.

Returns:

true if the proposed contact name was valid, or false if it was not.

Definition at line 338 of file contacts.c.

References chr_printable(), and st_empty_out().

Referenced by portal_endpoint_contacts_add().

5.177.2.23 int_t contacts_fetch (uint64_t usernum, contact_folder_t *folder)

Retrieve all of a user's contact entries in a specified contacts folder from the database.

Parameters:

usernum the numerical id of the user whose contacts will be retrieved.

folder a pointer to the contact folder which will have its contents listed.

Returns:

-1 on failure or 1 on success.

LOW: Should we bother with error checking?

Definition at line 339 of file datatier.c.

References contact_alloc(), contact_details_fetch(), contact_free(), contact_t, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_insert(), log_info, log_pedantic, M_TYPE_UINT64, mm_wipe(), PLACER, res_field_block(), res_field_length(), res_field_uint64(), res_row_next(), res_table_free(), stmt_get_result(), multi_t::u64, and multi_t::val.

Referenced by contacts_update().

5.177.2.24 inx_t* contacts_update (uint64_t usernum)

Retrieve all of a user's contacts (and contact folders) from the database.

Parameters:

usernum the numerical id of the user whose contact folders and contact entries should be fetched.

Returns:

NULL on failure, or a pointer to an inx holder containing all of the specified user's contact folders and entries on success.

Definition at line 152 of file contacts.c.

References contacts_fetch(), inx_cleanup(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_free(), inx_t, log_pedantic, M_FOLDER_CONTACTS, and magma_folder_fetch().

Referenced by meta_update_contacts().

5.177.3 Variable Documentation

5.177.3.1 contact_detail_t

Definition at line 19 of file contacts.h.

Referenced by contact_detail_alloc(), contact_details_fetch(), contact_edit(), contact_find_detail(), portal_contact_details(), portal_endpoint_contacts_copy(), and portal_endpoint_contacts_list().

5.177.3.2 `contact_t`

Definition at line 25 of file `contacts.h`.

Referenced by `contact_alloc()`, `contact_create()`, `contact_find_detail()`, `contact_find_name()`, `contact_find_number()`, `contact_move()`, `contacts_fetch()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, and `portal_endpoint_contacts_remove()`.

5.178 src/objects/contacts/find.c File Reference

```
#include "magma.h"
```

Functions

- `contact_t * contact_find_detail (contact_folder_t *folder, stringer_t *key, stringer_t *target)`
find.c
- `contact_t * contact_find_name (contact_folder_t *folder, stringer_t *target)`
Get a contact entry by name (case insensitive).
- `contact_t * contact_find_number (contact_folder_t *folder, uint64_t target)`
Find a contact entry in a contact folder by number.

5.178.1 Function Documentation

5.178.1.1 `contact_t* contact_find_detail (contact_folder_t * folder, stringer_t * key, stringer_t * target)`

find.c

Note:

This is a bit of a weird function that currently isn't being called from anywhere. It may or may not be here to stay.

Definition at line 13 of file find.c.

References `contact_detail_t`, `contact_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_find()`, `M_TYPE_STRINGER`, `st_cmp_ci_eq()`, and `st_empty`.

5.178.1.2 `contact_t* contact_find_name (contact_folder_t * folder, stringer_t * target)`

Get a contact entry by name (case insensitive).

Parameters:

- folder* a pointer to the contact folder to have its contents searched for the entry.
- target* a managed string containing the name of the target entry.

Returns:

NULL on failure or a pointer to the found user contact entry on success.

Definition at line 50 of file find.c.

References `contact_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `st_cmp_ci_eq()`, and `st_empty`.

5.178.1.3 `contact_t* contact_find_number (contact_folder_t * folder, uint64_t target)`

Find a contact entry in a contact folder by number.

Parameters:

- folder* a pointer to a contact folder object containing the contact entries to be searched.

target the numerical id of the contact entry to be located in the specified folder.

Returns:

NULL on failure or a pointer to the found user contact entry on success.

Definition at line 79 of file find.c.

References `contact_t`, `inx_find()`, `log_pedantic`, and `M_TYPE_UINT64`.

Referenced by `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_move()`, and `portal_endpoint_contacts_remove()`.

5.179 src/objects/folders/find.c File Reference

```
#include "magma.h"
```

Functions

- `magma_folder_t * magma_folder_find_name (inx_t *folders, stringer_t *target, uint64_t parent, bool_t check_inbox)`
Get a magma folder by name.
- `magma_folder_t * magma_folder_find_full_name (inx_t *folders, stringer_t *target, bool_t check_inbox)`
Get a magma folder by its fully qualified name.
- `magma_folder_t * magma_folder_find_number (inx_t *folders, uint64_t target)`
Get a magma folder by number.

5.179.1 Function Documentation

5.179.1.1 magma_folder_t* magma_folder_find_full_name (inx_t * *folders*, stringer_t * *target*, bool_t *check_inbox*)

Get a magma folder by its fully qualified name.

See also:

[magma_folder_name\(\)](#)

Parameters:

- folders* an inx holder containing the collection of magma folders to be searched.
target a managed string containing the fully qualified (expanded) name of the magma folder to be found.

Returns:

NULL on failure, or a pointer to the found folder on success.

Definition at line 56 of file find.c.

References [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [magma_folder_name\(\)](#), [magma_folder_t](#), [PLACER](#), [st_cleanup](#), [st_cmp_ci_eq\(\)](#), [st_cmp_cs_eq\(\)](#), and [st_empty](#).

5.179.1.2 magma_folder_t* magma_folder_find_name (inx_t * *folders*, stringer_t * *target*, uint64_t *parent*, bool_t *check_inbox*)

Get a magma folder by name. find.c

Parameters:

- folders* an inx holder containing the collection of magma folders to be searched.
target a managed string containing the name of the magma folder to be found.
parent the numerical id of the parent folder in which to search for the specified magma folder.
check_inbox if true, a case-insensitive comparison will be used if the specified folder name is "Inbox".

Returns:

NULL on failure, or a pointer to the found folder on success.

Definition at line 18 of file find.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `magma_folder_t`, `PLACER`, `st_cmp_ci_eq()`, `st_cmp_cs_eq()`, and `st_empty`.

Referenced by `contact_folder_create()`, `contact_folder_rename()`, and `portal_folder_contacts_add()`.

5.179.1.3 magma_folder_t* magma_folder_find_number (inx_t **folders*, uint64_t *target*)

Get a magma folder by number.

Parameters:

folders an inx holder containing the collection of magma folders to be searched.

target the numerical id of the magma folder to be found.

Returns:

NULL on failure, or a pointer to the found folder on success.

Definition at line 95 of file find.c.

References `inx_find()`, and `M_TYPE_UINT64`.

Referenced by `contact_folder_create()`, `magma_folder_name()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_folders_rename()`, `portal_folder_contacts_add()`, and `portal_folder_contacts_remove()`.

5.180 src/objects/folders/folders.c File Reference

```
#include "magma.h"
```

Functions

- `stringer_t * magma_folder_name (inx_t *folders, magma_folder_t *target)`
TODO: The assumption is that message folders will start using the interface as well.
- `int_t magma_folder_children (inx_t *folders, uint64_t target)`
Get the number of child folders a specified folder has.
- `void magma_folder_free (magma_folder_t *folder)`
Free a magma folder object and its underlying records.
- `magma_folder_t * magma_folder_alloc (uint64_t foldernum, uint64_t parent, uint32_t order, stringer_t *name)`
Create a new magma folder object.
- `bool_t magma_folder_funcs (uint_t type, magma_folder_t *(*folder_alloc)(uint64_t, uint64_t, uint32_t, stringer_t *), void(**folder_free)(magma_folder_t *))`
Retrieve the appropriate folder management functions for message folders or contact folders.

5.180.1 Function Documentation

5.180.1.1 magma_folder_t* magma_folder_alloc (uint64_t foldernum, uint64_t parent, uint32_t order, stringer_t * name)

Create a new magma folder object. folders.c

Parameters:

- foldernum*** the numerical id of this folder.
- parent*** the folder id of this folder's containing parent folder.
- order*** the order number of this folder in its parent folder.
- name*** a managed string with the name of the target folder.

Returns:

NULL on failure or a pointer to the newly allocated magma folder object on success.

Definition at line 103 of file folders.c.

References `align()`, `FOREIGNDATA`, `JOINTED`, `log_pedantic`, `magma_folder_t`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `PLACER_T`, `placer_t`, `rwlock_init()`, `st_data_get()`, `st_length_get()`, and `STACK`.

Referenced by `contact_folder_alloc()`, and `message_folder_alloc()`.

5.180.1.2 int_t magma_folder_children (inx_t * folders, uint64_t target)

Get the number of child folders a specified folder has.

Parameters:

- folders*** an inx object holding all of the user's folders.

target the folder number of the target folder.

Returns:

the number of child folders contained by the specified folder, or 0 on failure.

Definition at line 57 of file folders.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, and `magma_folder_t`.

Referenced by `contact_folder_create()`, `contact_folder_remove()`, `message_folder_create()`, and `message_folder_remove()`.

5.180.1.3 void magma_folder_free (magma_folder_t * *folder*)

Free a magma folder object and its underlying records.

Returns:

This function returns no value.

Definition at line 84 of file folders.c.

References `inx_cleanup()`, `mm_free()`, and `rwlock_destroy()`.

Referenced by `contact_folder_free()`, and `message_folder_free()`.

5.180.1.4 bool_t magma_folder_funcs (uint_t *type*, magma_folder_t *(*)(uint64_t, uint64_t, uint32_t, stringer_t *) *folder_alloc*, void(**)(magma_folder_t *) *folder_free*)

Retrieve the appropriate folder management functions for message folders or contact folders.

Parameters:

type the magma folder class of the desired management functions (M_FOLDER_MESSAGES or M_FOLDER_CONTACTS).

folder_alloc the address of a folder allocation function pointer that will be updated appropriately.

folder_free the address of a folder free function pointer that will be updated appropriately.

Returns:

true on success or false on failure.

Definition at line 136 of file folders.c.

References `contact_folder_alloc()`, `contact_folder_free()`, `M_FOLDER_CONTACTS`, `M_FOLDER_MESSAGES`, `message_folder_alloc()`, and `message_folder_free()`.

Referenced by `magma_folder_fetch()`.

5.180.1.5 stringer_t* magma_folder_name (inx_t * *folders*, magma_folder_t * *target*)

TODO: The assumption is that message folders will start using the interface as well. Get the fully expanded name of a folder.

Note:

The folder hierarchy will be delimited with a "." character.

Parameters:

folders the inx object holding all of a user's magma folders.

target a pointer to the magma folder object to have its name expanded.

Returns:

NULL on failure or a pointer to a managed string with the fully expanded name of the specified folder.

Definition at line 19 of file folders.c.

References FOLDER_RECURSION_LIMIT, HEAP, JOINTED, log_pedantic, magma_folder_find_number(), MANAGED_T, PLACER, st_append, st_dupe_opts(), and st_empty.

Referenced by imap_list(), magma_folder_find_full_name(), and portal_endpoint_folders_rename().

5.181 src/objects/meta/folders.c File Reference

```
#include "magma.h"
```

Functions

- [meta_stats_tag_t](#) * [meta_folder_stats_tag_alloc](#) ([stringer_t](#) *tag)
Allocate a new meta tag stat object for tracking message tags.
- [inx_t](#) * [meta_folders_stats_tags](#) ([inx_t](#) *messages, [uint64_t](#) folder)
Create a collection of meta tag stats for a set of messages.
- [stringer_t](#) * [meta_folders_name](#) ([inx_t](#) *list, [meta_folder_t](#) *folder)
Get a folder's fully qualified name.
- [meta_folder_t](#) * [meta_folders_by_name](#) ([inx_t](#) *folders, [stringer_t](#) *name)
Get a folder by its fully qualified name.
- [meta_folder_t](#) * [meta_folders_by_number](#) ([inx_t](#) *folders, [uint64_t](#) number)
Get a folder by number.
- [int_t](#) [meta_folders_children](#) ([inx_t](#) *folders, [uint64_t](#) number)
Get the number of direct child folders of a specified parent folder.

5.181.1 Function Documentation

5.181.1.1 [meta_stats_tag_t](#)* [meta_folder_stats_tag_alloc](#) ([stringer_t](#) * tag)

Allocate a new meta tag stat object for tracking message tags. folders.c

Parameters:

tag a managed string containing the name of the message tag.

Returns:

NULL on failure, or a newly allocated and initialized meta tag stat object on success.

Definition at line 15 of file folders.c.

References [align\(\)](#), [FOREIGNDATA](#), [JOINTED](#), [log_pedantic](#), [mm_alloc\(\)](#), [mm_copy\(\)](#), [PLACER_T](#), [placer_t](#), [st_data_get\(\)](#), [st_length_get\(\)](#), [STACK](#), and [meta_stats_tag_t::tag](#).

Referenced by [meta_folders_stats_tags\(\)](#).

5.181.1.2 [meta_folder_t](#)* [meta_folders_by_name](#) ([inx_t](#) * folders, [stringer_t](#) * name)

Get a folder by its fully qualified name.

Parameters:

folders a pointer to an inx holder containing a list of folders to be traversed in the search.

folder a pointer to the meta folder object that is the leaf node of the folder path.

Returns:

NULL on failure or a managed string containing the fully qualified folder name on success.

Definition at line 138 of file folders.c.

References `CONSTANT`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `meta_folder_t`, `meta_folders_name()`, `st_cmp_ci_eq()`, `st_cmp_cs_eq()`, and `st_free()`.

Referenced by `imap_append()`, `imap_copy()`, `imap_folder_create()`, `imap_folder_remove()`, `imap_folder_rename()`, `imap_folder_status()`, `imap_subscribe()`, and `portal_folder_mail_add()`.

5.181.1.3 meta_folder_t* meta_folders_by_number (inx_t * *folders*, uint64_t *number*)

Get a folder by number.

Parameters:

folders a pointer to an `inx` holder containing the folders to be searched.

number the numerical id of the folder to be retrieved.

Returns:

NULL on failure, or a pointer to the found meta folder object of the specified folder on success.

Definition at line 179 of file folders.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, and `meta_folder_t`.

Referenced by `meta_folders_name()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_tag()`, `portal_folder_mail_add()`, and `portal_folder_mail_remove()`.

5.181.1.4 int_t meta_folders_children (inx_t * *folders*, uint64_t *number*)

Get the number of direct child folders of a specified parent folder.

Parameters:

folders a pointer to an `inx` holder containing the the collection of folders to be traversed.

number the folder id of the parent folder to be scanned for children.

Returns:

the number of children folders in the specified parent folder, or 0 on failure.

Definition at line 207 of file folders.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, and `meta_folder_t`.

Referenced by `imap_folder_remove()`.

5.181.1.5 stringer_t* meta_folders_name (inx_t * *list*, meta_folder_t * *folder*)

Get a folder's fully qualified name.

Note:

This function will prepend the entire ancestor path of a folder to its name, with each level delimited by periods.

Parameters:

list a pointer to an inx holder containing a list of folders to be searched for parent folders.

folder a pointer to the meta folder object that is the leaf node of the folder path.

Returns:

NULL on failure or a managed string containing the fully qualified folder name on success.

Definition at line 99 of file folders.c.

References FOLDER_RECURSION_LIMIT, log_pedantic, meta_folders_by_number(), ns_length_get(), st_free(), st_import(), and st_merge.

Referenced by imap_folder_name_escaped(), imap_narrow_folders(), meta_folders_by_name(), portal_endpoint_folders_rename(), portal_folder_mail_add(), and portal_folder_mail_remove().

5.181.1.6 inx_t* meta_folders_stats_tags (inx_t * *messages*, uint64_t *folder*)

Create a collection of meta tag stats for a set of messages.

Note:

Each meta tag stat will contain the name of a message tag, along with the number of messages with which it was associated.

Parameters:

messages an inx holder containing the list of messages to be examined.

folder the numerical id of the parent folder that contains all target messages.

Returns:

NULL on failure, or an inx holder containing all of the messages' meta tag stats on success.

LOW: This is a very inefficient method for counting the number of times each tag appears.

Definition at line 40 of file folders.c.

References ar_field_st(), ar_length_get(), meta_stats_tag_t::count, inx_alloc(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_find(), inx_insert(), inx_t, log_pedantic, M_INX_HASHED, M_TYPE_STRINGER, meta_folder_stats_tag_alloc(), meta_message_t, mm_free(), multi_t::st, and multi_t::val.

Referenced by portal_endpoint_folders_tags().

5.182 src/servers/imap/folders.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * imap_folder_name_escaped](#) ([inx_t](#) *folders, [meta_folder_t](#) *active)
Get an imap folder's escaped name.
- [uint64_t imap_next_folder_order](#) ([inx_t](#) *folders, [uint64_t](#) parent)
Get the next order value for an imap folder.
- [int_t imap_folder_compare](#) ([stringer_t](#) *name, [stringer_t](#) *compare)
- [bool_t imap_valid_folder_name](#) ([stringer_t](#) *name)
Check to see if a name is a valid imap folder name.
- [int_t imap_count_folder_levels](#) ([stringer_t](#) *name)
Get the highest node level indicated by an imap folder name.
- [int_t imap_folder_create](#) ([uint64_t](#) usernum, [inx_t](#) *folders, [stringer_t](#) *name)
Create a new imap folder.
- [int_t imap_folder_remove](#) ([uint64_t](#) usernum, [inx_t](#) *folders, [inx_t](#) *messages, [stringer_t](#) *name)
Remove an imap folder, both in memory and on the database.
- [int_t imap_folder_rename](#) ([uint64_t](#) usernum, [inx_t](#) *folders, [stringer_t](#) *original, [stringer_t](#) *rename)
Rename an imap folder.
- [int_t imap_folder_status](#) ([inx_t](#) *folders, [inx_t](#) *messages, [stringer_t](#) *name, [imap_folder_status_t](#) *status)
Get the status of a folder.
- [inx_t * imap_narrow_folders](#) ([inx_t](#) *folders, [stringer_t](#) *reference, [stringer_t](#) *mailbox)

5.182.1 Function Documentation

5.182.1.1 [int_t imap_count_folder_levels](#) ([stringer_t](#) * name)

Get the highest node level indicated by an imap folder name. folders.c

Note:

The smallest possible node level value is 1.

Parameters:

name a managed string containing the name of the imap folder.

Returns:

0 on failure, or the number of node levels indicated by the specified imap folder name, on success.

Definition at line 218 of file folders.c.

References [IMAP_FOLDER_RECURSION_LMIIT](#), [log_pedantic](#), and [st_empty_out\(\)](#).

Referenced by [imap_folder_create\(\)](#), and [imap_folder_rename\(\)](#).

5.182.1.2 `int_t imap_folder_compare (stringer_t * name, stringer_t * compare)`

Definition at line 91 of file folders.c.

References PLACER, st_char_get(), st_cmp_ci_eq(), and st_length_get().

Referenced by imap_narrow_folders().

5.182.1.3 `int_t imap_folder_create (uint64_t usernum, inx_t * folders, stringer_t * name)`

Create a new imap folder.

Note:

If they don't already exist, any parent folders specified in the fully qualified folder name will automatically be created first.

Parameters:

usenum the numerical id of the user to whom the new imap folder will belong.

folders an inx holder containing the set of folders into which the new imap folder will be inserted.

name a managed string containing the fully qualified name of the new imap folder to be created.

Returns:

1 on success or <= 0 on failure. 0: An invalid parameter was passed to the function, or an internal failure occurred. -1: The specified new folder name was invalid. -2: The node depth of the new folder is too large (imap folder recursion limit reached [IMAP_FOLDER_RECURSION_LMIIT]). -3: Special folder "Inbox" cannot contain subfolders. -4: Part of the folder path name exceeds 16 characters (FOLDER_LENGTH_LIMIT) after being unescaped for quotation marks. -5: The specified folder name already exists.

Definition at line 264 of file folders.c.

References CONTIGUOUS, FOLDER_LENGTH_LIMIT, HEAP, imap_count_folder_levels(), IMAP_FOLDER_RECURSION_LMIIT, imap_next_folder_order(), imap_valid_folder_name(), inx_insert(), log_pedantic, M_TYPE_UINT64, MANAGED_T, meta_data_insert_folder(), meta_folder_t, meta_folders_by_name(), mm_alloc(), mm_copy(), mm_free(), pl_data_get(), pl_length_get(), PLACER, placer_t, st_cleanup, st_cmp_ci_eq(), st_dupe(), st_dupe_opts(), st_free(), st_merge, st_replace(), tok_get_st(), multi_t::u64, and multi_t::val.

Referenced by imap_create(), imap_subscribe(), and portal_folder_mail_add().

5.182.1.4 `stringer_t* imap_folder_name_escaped (inx_t * folders, meta_folder_t * active)`

Get an imap folder's escaped name. TODO: Since the Portal needs access to some of these folder manipulation functions the common logic should be extracted and moved into the folder object interface.

Parameters:

folders an inx holder containing the ancestor folders of the specified imap folder.

active a pointer to the meta folder object of the folder to have its name escaped.

Returns:

NULL on failure, or a pointer to a managed string containing the escaped name of the specified imap folder on success.

Definition at line 19 of file folders.c.

References meta_folders_name(), PLACER, st_free(), st_merge, and st_replace().

Referenced by imap_list(), and imap_lsub().

5.182.1.5 int_t imap_folder_remove (uint64_t *usernum*, inx_t * *folders*, inx_t * *messages*, stringer_t * *name*)

Remove an imap folder, both in memory and on the database.

Note:

Any child messages of the specified folder will be deleted, but if it has child folders, the folder will not be deleted. The function also protects the special "Inbox" folder from deletion.

Parameters:

usernum the numerical id of the user that owns the imap folder to be deleted.

folders an inx holder containing a collection of folders for looking up the specified imap folder by name.

messages an inx holder containing a collection of messages for looking up the contents of the specified imap folder.

name a managed string containing the name of the imap folder to be deleted.

Returns:

<= 0 on failure, or 1 on success. 0: General failure or if there was an error removing the folder from the database. -1: The specified folder name was invalid. -2: Removal failed because the "Inbox" folder was specified. -3: The specified folder could not be found.

Definition at line 412 of file folders.c.

References `imap_valid_folder_name()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_delete()`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `mail_remove_message()`, `meta_data_delete_folder()`, `meta_folder_t`, `meta_folders_by_name()`, `meta_folders_children()`, `meta_message_t`, `PLACER`, `placer_t`, `st_cmp_ci_eq()`, `tok_get_st()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_delete()`, and `portal_folder_mail_remove()`.

5.182.1.6 int_t imap_folder_rename (uint64_t *usernum*, inx_t * *folders*, stringer_t * *original*, stringer_t * *rename*)

Rename an imap folder.

Note:

Any parent folder components of the folder's fully qualified name will be created if they do not already exist.

Parameters:

usernum the numerical id of the user requesting the folder renaming.

folders an inx holder containing the folders to be searched for the folder specified to be renamed.

original a managed string containing the original name of the folder to be renamed.

rename a managed string containing the new name of the specified folder.

Returns:

1 on success or 0 or less on failure. 0: One of the parameters passed to the function was invalid, or there was a memory allocation failure. -1: Either the original or new folder name was invalid. -2: Was unable to rename the "Inbox" folder. -3: The specified imap folder did not exist. -4: Either the original or new name exceeded the imap folder recursion limit. -5: A folder already exists with the new folder name. -6: A segment of the folder name was larger than `FOLDER_LENGTH_LIMIT` (16 bytes).

Definition at line 488 of file folders.c.

References `CONTIGUOUS`, `FOLDER_LENGTH_LIMIT`, `HEAP`, `imap_count_folder_levels()`, `IMAP_FOLDER_RECURSION_LMIIT`, `imap_next_folder_order()`, `imap_valid_folder_name()`, `inx_insert()`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `MANAGED_T`, `meta_data_insert_folder()`, `meta_data_update_folder_name()`, `meta_folder_t`, `meta_folders_by_name()`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `mm_wipe()`, `pl_data_get()`, `pl_length_get()`, `PLACER`, `placer_t`, `st_cleanup`, `st_cmp_ci_eq()`, `st_dupe()`, `st_dupe_opts()`, `st_free()`, `st_merge`, `st_replace()`, `tok_get_st()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_rename()`, and `portal_endpoint_folders_rename()`.

5.182.1.7 `int_t imap_folder_status (inx_t * folders, inx_t * messages, stringer_t * name, imap_folder_status_t * status)`

Get the status of a folder.

Note:

This function will count the number of messages in a folder, as well as the number of messages marked recent or unseen, as well as the numerical id of the first message in the folder and the UIDNEXT of the specified folder.

Parameters:

folders an inx holder containing a list of folders to be searched for the specified folder.

messages an inx holder containing a complete list of a user's messages to be examined for gathering statistics.

name a managed string containing the name of the imap folder to be queried.

status a pointer to an imap folder status object to receive the folder's status information.

Returns:

1 on success or ≤ 0 on failure. 0: General failure. -1: The specified folder name was invalid. -2: The folder did not exist.

Definition at line 678 of file folders.c.

References `imap_folder_status_t::first`, `imap_folder_status_t::foldernum`, `imap_valid_folder_name()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `imap_folder_status_t::messages`, `meta_folder_t`, `meta_folders_by_name()`, `meta_message_t`, `mm_wipe()`, `imap_folder_status_t::recent`, `imap_folder_status_t::uidnext`, and `imap_folder_status_t::unseen`.

Referenced by `imap_examine()`, `imap_select()`, and `imap_status()`.

5.182.1.8 `inx_t* imap_narrow_folders (inx_t * folders, stringer_t * reference, stringer_t * mailbox)`

Definition at line 743 of file folders.c.

References `imap_folder_compare()`, `inx_alloc()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_insert()`, `inx_t`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_folder_t`, `meta_folders_name()`, `mm_dupe()`, `mm_free()`, `st_free()`, `st_length_get()`, `st_merge`, `multi_t::u64`, and `multi_t::val`.

Referenced by `imap_list()`, and `imap_lsub()`.

5.182.1.9 `uint64_t imap_next_folder_order (inx_t * folders, uint64_t parent)`

Get the next order value for an imap folder.

Note:

The next order value is the current highest order value of any child folder in the specified directory, incremented by one.

Parameters:

folder an inx holder containing the collection of imap folders to be scanned.

parent the numerical id of the folder to be queried.

Returns:

0 on failure, or the next order value of the specified folder on success.

Definition at line 67 of file folders.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, and `meta_folder_t`.

Referenced by `imap_folder_create()`, and `imap_folder_rename()`.

5.182.1.10 `bool_t imap_valid_folder_name (stringer_t * name)`

Check to see if a name is a valid imap folder name.

Note:

The following naming checks are enforced: 1. The name can't be empty or begin with a period. 2. It cannot contain a non-printable character. 3. Trailing periods will be removed. 4. The folder name cannot contain consecutive periods.

Parameters:

name a managed string containing the imap folder name to be validated.

Returns:

true on success or false on failure.

Definition at line 157 of file folders.c.

References `log_pedantic`, `st_empty_out()`, `st_length_get()`, and `st_length_set()`.

Referenced by `contact_folder_create()`, `contact_folder_rename()`, `imap_folder_create()`, `imap_folder_remove()`, `imap_folder_rename()`, and `imap_folder_status()`.

5.183 src/web/portal/folders.c File Reference

```
#include "magma.h"
```

Functions

- void `portal_folder_mail_add` (`connection_t` *con, `stringer_t` *name, uint64_t parent)
Add a new mail folder for a user.
- void `portal_folder_contacts_add` (`connection_t` *con, `stringer_t` *name, uint64_t parent)
Add a new contact folder for a user.
- void `portal_folder_mail_remove` (`connection_t` *con, uint64_t foldernum)
Remove a user's mail folder.
- void `portal_folder_contacts_remove` (`connection_t` *con, uint64_t foldernum)
Remove a user's contacts folder.

5.183.1 Function Documentation

5.183.1.1 void portal_folder_contacts_add (connection_t * con, stringer_t * name, uint64_t parent)

Add a new contact folder for a user. folders.c

Note:

If parent is 0, then the new folder is assumed to be a root folder. This function returns no value, but returns the appropriate portal json-rpc response directly.

Parameters:

- con** a pointer to the connection object across which the add folder response will be sent.
- name** a managed string containing the name of the new folder.
- parent** the numerical id of the folder that will be the parent of the new folder (or 0 to specify a root folder).

Returns:

This function returns no value.

Definition at line 82 of file folders.c.

References `contact_folder_create()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `magma_folder_find_name()`, `magma_folder_find_number()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_FOLDERS_ADD`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `sess_serial_check()`, and `st_cleanup`.

Referenced by `portal_endpoint_folders_add()`.

5.183.1.2 void portal_folder_contacts_remove (connection_t * con, uint64_t foldernum)

Remove a user's contacts folder.

Parameters:

- con** a pointer to the connection object across which the remove folder response will be sent.

foldernum the numerical id of the contacts folder that will be removed.

Returns:

This function returns no value.

Definition at line 201 of file folders.c.

References `contact_folder_remove()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `magma_folder_find_number()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION`, `PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, and `sess_serial_check()`.

Referenced by `portal_endpoint_folders_remove()`.

5.183.1.3 void portal_folder_mail_add (connection_t * con, stringer_t * name, uint64_t parent)

Add a new mail folder for a user.

Note:

If parent is 0, then the new folder is assumed to be a root folder. This function returns no value, but returns the appropriate portal json-rpc response directly.

Parameters:

con a pointer to the connection object across which the add folder response will be sent.

name a managed string containing the name of the new folder.

parent the numerical id of the folder that will be the parent of the new folder (or 0 to specify a root folder).

Returns:

This function returns no value.

Definition at line 20 of file folders.c.

References `imap_folder_create()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `meta_folder_t`, `meta_folders_by_name()`, `meta_folders_by_number()`, `meta_folders_name()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_FOLDERS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_FOLDERS_ADD`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `sess_serial_check()`, `st_cleanup`, and `st_merge`.

Referenced by `portal_endpoint_folders_add()`.

5.183.1.4 void portal_folder_mail_remove (connection_t * con, uint64_t foldernum)

Remove a user's mail folder.

Parameters:

con a pointer to the connection object across which the remove folder response will be sent.

foldernum the numerical id of the mail folder that will be removed.

Returns:

This function returns no value.

Definition at line 148 of file folders.c.

References `imap_folder_remove()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `meta_folder_t`, `meta_folders_by_number()`, `meta_folders_name()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_FOLDERS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION`, `PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `sess_serial_check()`, and `st_cleanup`.

Referenced by `portal_endpoint_folders_remove()`.

5.184 src/objects/folders/folders.h File Reference

Defines

- #define [FOLDER_LENGTH_LIMIT](#) 16
- #define [FOLDER_RECURSION_LIMIT](#) 16

Typedefs

- typedef [magma_folder_t](#) [contact_folder_t](#)
- typedef [magma_folder_t](#) [message_folder_t](#)

Enumerations

- enum { [M_FOLDER_MESSAGES](#) = 1, [M_FOLDER_CONTACTS](#) = 2 }

Functions

- struct [__attribute__](#) ((packed))
- [message_folder_t](#) * [message_folder_alloc](#) (uint64_t foldernum, uint64_t parent, uint32_t order, [stringer_t](#) *name)
messages.c
- [int_t](#) [message_folder_create](#) (uint64_t usernum, uint64_t parent, [stringer_t](#) *name, [inx_t](#) *folders)
- void [message_folder_free](#) ([message_folder_t](#) *folder)
Free a message folder object.
- [bool_t](#) [message_folder_remove](#) (uint64_t usernum, uint64_t foldernum, [inx_t](#) *folders)
Remove a user's message folder.
- [contact_folder_t](#) * [contact_folder_alloc](#) (uint64_t foldernum, uint64_t parent, uint32_t order, [stringer_t](#) *name)
contacts.c
- [int_t](#) [contact_folder_create](#) (uint64_t usernum, uint64_t parent, [stringer_t](#) *name, [inx_t](#) *folders)
Create a new contact folder.
- void [contact_folder_free](#) ([contact_folder_t](#) *folder)
Free a contact folder object.
- [int_t](#) [contact_folder_remove](#) (uint64_t usernum, uint64_t foldernum, [inx_t](#) *folders)
Remove a user's contact folder.
- [bool_t](#) [contact_folder_rename](#) (uint64_t usernum, uint64_t foldernum, [stringer_t](#) *rename, [inx_t](#) *folders)
Rename a user's contact folder.
- [bool_t](#) [magma_folder_delete](#) (uint64_t usernum, uint64_t foldernum, [uint_t](#) type)
datatier.c
- [inx_t](#) * [magma_folder_fetch](#) (uint64_t usernum, [uint_t](#) type)
Fetch all of a user's folders of a specified type from the database.
- uint64_t [magma_folder_insert](#) (uint64_t usernum, [stringer_t](#) *name, uint64_t parent, uint32_t order, [uint_t](#) type)
Insert a new folder into the database.

- [bool_t magma_folder_rename](#) (uint64_t usernum, uint64_t foldernum, [uint_t](#) type, [stringer_t](#) *rename)
Rename a folder in the database.
- [magma_folder_t * magma_folder_find_name](#) ([inx_t](#) *folders, [stringer_t](#) *target, uint64_t parent, [bool_t](#) check_inbox)
find.c
- [magma_folder_t * magma_folder_find_full_name](#) ([inx_t](#) *folders, [stringer_t](#) *target, [bool_t](#) check_inbox)
Get a magma folder by its fully qualified name.
- [magma_folder_t * magma_folder_find_number](#) ([inx_t](#) *folders, uint64_t target)
Get a magma folder by number.
- [magma_folder_t * magma_folder_alloc](#) (uint64_t foldernum, uint64_t parent, uint32_t order, [stringer_t](#) *name)
folders.c
- [int_t magma_folder_children](#) ([inx_t](#) *folders, uint64_t target)
Get the number of child folders a specified folder has.
- void [magma_folder_free](#) ([magma_folder_t](#) *folder)
Free a magma folder object and its underlying records.
- [bool_t magma_folder_funcs](#) ([uint_t](#) type, [magma_folder_t](#) *(*folder_alloc)(uint64_t, uint64_t, uint32_t, [stringer_t](#) *), void(**folder_free)([magma_folder_t](#) *))
Retrieve the appropriate folder management functions for message folders or contact folders.
- [stringer_t * magma_folder_name](#) ([inx_t](#) *folders, [magma_folder_t](#) *target)
TODO: The assumption is that message folders will start using the interface as well.

Variables

- [magma_folder_t](#)

5.184.1 Define Documentation

5.184.1.1 #define FOLDER_LENGTH_LIMIT 16

Definition at line 11 of file folders.h.

Referenced by [contact_folder_create\(\)](#), [contact_folder_rename\(\)](#), [imap_create\(\)](#), [imap_folder_create\(\)](#), [imap_folder_rename\(\)](#), [imap_rename\(\)](#), and [imap_subscribe\(\)](#).

5.184.1.2 #define FOLDER_RECURSION_LIMIT 16

Definition at line 12 of file folders.h.

Referenced by [magma_folder_name\(\)](#), and [meta_folders_name\(\)](#).

5.184.2 Typedef Documentation

5.184.2.1 typedef magma_folder_t contact_folder_t

Definition at line 30 of file folders.h.

5.184.2.2 typedef magma_folder_t message_folder_t

Definition at line 31 of file folders.h.

5.184.3 Enumeration Type Documentation

5.184.3.1 anonymous enum

Enumerator:

M_FOLDER_MESSAGES M_FOLDER_MESSAGES.

M_FOLDER_CONTACTS M_FOLDER_CONTACTS.

Definition at line 17 of file folders.h.

5.184.4 Function Documentation

5.184.4.1 struct __attribute__((packed)) [read]

Definition at line 22 of file folders.h.

References *inx_t*, and *lock*.

5.184.4.2 contact_folder_t* contact_folder_alloc (uint64_t foldernum, uint64_t parent, uint32_t order, stringer_t * name)

contacts.c contacts.c

Parameters:

foldernum the numerical id of this contact folder.

parent the numerical id of the parent folder of this contact folder.

order the order number of this folder in its parent folder.

name a managed string containing the name of this folder.

Returns:

NULL on failure, or a pointer to the newly initialized contact folder object on success.

Definition at line 29 of file contacts.c.

References *contact_free()*, *inx_alloc()*, *M_INX_TREE*, *magma_folder_alloc()*, and *mm_cleanup*.

Referenced by *contact_folder_create()*, and *magma_folder_funcs()*.

5.184.4.3 int_t contact_folder_create (uint64_t usernum, uint64_t parent, stringer_t * name, inx_t * folders)

Create a new contact folder.

Parameters:

usernum the numerical id of the user to whom the new imap folder will belong.

parent the numerical id of the parent folder to contain the new contact folder, or 0 for a root folder.

name a managed string containing the fully qualified name of the new contact folder to be created.

folders an *inx* holder containing the set of folders into which the new contact folder will be inserted.

Returns:

1 on success or ≤ 0 on failure. 0: An invalid parameter was passed to the function, or an internal failure occurred. -1: The specified new folder name was invalid. -2: Part of the folder path name exceeds 16 characters (FOLDER_LENGTH_LIMIT) after being unescaped for quotation marks. -3: The specified folder name already exists. -4: An invalid parent folder was specified. -5: Part of the folder path name exceeds 16 characters (FOLDER_LENGTH_LIMIT) after being unescaped for quotation marks.

Definition at line 55 of file contacts.c.

References `contact_folder_alloc()`, `contact_folder_free()`, `FOLDER_LENGTH_LIMIT`, `IMAP_FOLDER_RECURSION_LMIIT`, `imap_valid_folder_name()`, `inx_insert()`, `log_pedantic`, `M_FOLDER_CONTACTS`, `M_TYPE_UINT64`, `magma_folder_children()`, `magma_folder_find_name()`, `magma_folder_find_number()`, `magma_folder_insert()`, `magma_folder_t`, `st_empty`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_folder_contacts_add()`.

5.184.4.4 void contact_folder_free (contact_folder_t *folder)

Free a contact folder object.

Parameters:

folder a pointer to the contact folder object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file contacts.c.

References `magma_folder_free()`.

Referenced by `contact_folder_create()`, and `magma_folder_funcs()`.

5.184.4.5 int_t contact_folder_remove (uint64_t usernum, uint64_t foldernum, inx_t *folders)

Remove a user's contact folder.

See also:

[magma_folder_delete\(\)](#)

Note:

This operation will only succeed if the specified contact folder is empty.

Parameters:

usenum the numerical id of the user requesting the contact folder removal.

foldernum the numerical id of the folder to be removed.

folders an inx holder containing the contact folders to be searched for the folder specified for removal.

Returns:

0 on success or < 0 on failure. -1: There was an unexpected internal error. -2: The specified folder could not be found. -3: Contact folder was not empty.

Definition at line 121 of file contacts.c.

References `inx_count()`, `inx_delete()`, `inx_find()`, `log_pedantic`, `M_FOLDER_CONTACTS`, `M_TYPE_UINT64`, `magma_folder_children()`, and `magma_folder_delete()`.

Referenced by `portal_folder_contacts_remove()`.

5.184.4.6 bool_t contact_folder_rename (uint64_t *usernum*, uint64_t *foldernum*, stringer_t * *rename*, inx_t * *folders*)

Rename a user's contact folder.

See also:

[magma_folder_rename\(\)](#)

Parameters:

usernum the numerical id of the user requesting the contact folder renaming.

foldernum the numerical id of the folder to be renamed.

rename a managed string containing the new name of the specified folder.

folders an inx holder containing the contact folders to be searched for the folder specified to be renamed.

Returns:

true on success or false on failure.

Definition at line 161 of file contacts.c.

References FOLDER_LENGTH_LIMIT, imap_valid_folder_name(), inx_find(), log_pedantic, M_FOLDER_CONTACTS, M_TYPE_UINT64, magma_folder_find_name(), magma_folder_rename(), st_dupe(), st_empty(), st_free(), and st_length_get().

Referenced by portal_endpoint_folders_rename().

5.184.4.7 magma_folder_t* magma_folder_alloc (uint64_t *foldernum*, uint64_t *parent*, uint32_t *order*, stringer_t * *name*)

folders.c folders.c

Parameters:

foldernum the numerical id of this folder.

parent the folder id of this folder's containing parent folder.

order the order number of this folder in its parent folder.

name a managed string with the name of the target folder.

Returns:

NULL on failure or a pointer to the newly allocated magma folder object on success.

Definition at line 103 of file folders.c.

References align(), FOREIGNDATA, JOINTED, log_pedantic, magma_folder_t, mm_alloc(), mm_copy(), mm_free(), PLACER_T, placer_t, rwlock_init(), st_data_get(), st_length_get(), and STACK.

Referenced by contact_folder_alloc(), and message_folder_alloc().

5.184.4.8 int_t magma_folder_children (inx_t * *folders*, uint64_t *target*)

Get the number of child folders a specified folder has.

Parameters:

folders an inx object holding all of the user's folders.

target the folder number of the target folder.

Returns:

the number of child folders contained by the specified folder, or 0 on failure.

Definition at line 57 of file folders.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, and `magma_folder_t`.

Referenced by `contact_folder_create()`, `contact_folder_remove()`, `message_folder_create()`, and `message_folder_remove()`.

5.184.4.9 `bool_t magma_folder_delete (uint64_t usernum, uint64_t foldernum, uint_t type)`

datatier.c datatier.c

Parameters:

usernum the numerical id of the user requesting the folder removal.

foldernum the numerical id of the folder to be removed.

type the type of the folder to be deleted (can be `M_FOLDER_MESSAGES` or `M_FOLDER_CONTACTS`).

Returns:

true on success or false on failure.

Definition at line 136 of file datatier.c.

References `mm_wipe()`, and `stmt_exec_affected()`.

Referenced by `contact_folder_remove()`, and `message_folder_remove()`.

5.184.4.10 `inx_t* magma_folder_fetch (uint64_t usernum, uint_t type)`

Fetch all of a user's folders of a specified type from the database.

Parameters:

usernum the numerical id of the requested user.

type the folder class of the folders to be retrieved (can be `M_FOLDER_MESSAGES` or `M_FOLDER_CONTACTS`).

Returns:

NULL on failure, or an `inx` object holding all of the requested folders on success.

Definition at line 16 of file datatier.c.

References `inx_alloc()`, `inx_free()`, `inx_insert()`, `inx_t`, `log_info`, `log_pedantic`, `M_INX_TREE`, `M_TYPE_UINT64`, `magma_folder_funcs()`, `magma_folder_t`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `contacts_update()`, and `messages_update()`.

5.184.4.11 `magma_folder_t* magma_folder_find_full_name (inx_t *folders, stringer_t *target, bool_t check_inbox)`

Get a magma folder by its fully qualified name.

See also:

[`magma_folder_name\(\)`](#)

Parameters:

folders an `inx` holder containing the collection of magma folders to be searched.

target a managed string containing the fully qualified (expanded) name of the magma folder to be found.

Returns:

NULL on failure, or a pointer to the found folder on success.

Definition at line 56 of file find.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `magma_folder_name()`, `magma_folder_t`, `PLACER`, `st_cleanup`, `st_cmp_ci_eq()`, `st_cmp_cs_eq()`, and `st_empty`.

5.184.4.12 magma_folder_t* magma_folder_find_name (inx_t * *folders*, stringer_t * *target*, uint64_t *parent*, bool_t *check_inbox*)

find.c find.c

Parameters:

- folders* an inx holder containing the collection of magma folders to be searched.
- target* a managed string containing the name of the magma folder to be found.
- parent* the numerical id of the parent folder in which to search for the specified magma folder.
- check_inbox* if true, a case-insensitive comparison will be used if the specified folder name is "Inbox".

Returns:

NULL on failure, or a pointer to the found folder on success.

Definition at line 18 of file find.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `magma_folder_t`, `PLACER`, `st_cmp_ci_eq()`, `st_cmp_cs_eq()`, and `st_empty`.

Referenced by `contact_folder_create()`, `contact_folder_rename()`, and `portal_folder_contacts_add()`.

5.184.4.13 magma_folder_t* magma_folder_find_number (inx_t * *folders*, uint64_t *target*)

Get a magma folder by number.

Parameters:

- folders* an inx holder containing the collection of magma folders to be searched.
- target* the numerical id of the magma folder to be found.

Returns:

NULL on failure, or a pointer to the found folder on success.

Definition at line 95 of file find.c.

References `inx_find()`, and `M_TYPE_UINT64`.

Referenced by `contact_folder_create()`, `magma_folder_name()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_folders_rename()`, `portal_folder_contacts_add()`, and `portal_folder_contacts_remove()`.

5.184.4.14 void magma_folder_free (magma_folder_t * *folder*)

Free a magma folder object and its underlying records.

Returns:

This function returns no value.

Definition at line 84 of file folders.c.

References `inx_cleanup()`, `mm_free()`, and `rwlock_destroy()`.

Referenced by `contact_folder_free()`, and `message_folder_free()`.

5.184.4.15 `bool_t magma_folder_funcs (uint_t type, magma_folder_t *(*)(uint64_t, uint64_t, uint32_t, stringer_t *) folder_alloc, void(**)(magma_folder_t *) folder_free)`

Retrieve the appropriate folder management functions for message folders or contact folders.

Parameters:

type the magma folder class of the desired management functions (M_FOLDER_MESSAGES or M_FOLDER_CONTACTS).

folder_alloc the address of a folder allocation function pointer that will be updated appropriately.

folder_free the address of a folder free function pointer that will be updated appropriately.

Returns:

true on success or false on failure.

Definition at line 136 of file folders.c.

References `contact_folder_alloc()`, `contact_folder_free()`, `M_FOLDER_CONTACTS`, `M_FOLDER_MESSAGES`, `message_folder_alloc()`, and `message_folder_free()`.

Referenced by `magma_folder_fetch()`.

5.184.4.16 `uint64_t magma_folder_insert (uint64_t usernum, stringer_t * name, uint64_t parent, uint32_t order, uint_t type)`

Insert a new folder into the database.

Parameters:

usernum the numerical id of the user to whom the new folder belongs.

name a managed string containing the name of the new folder.

parent the numerical id of the mail folder to be the parent of the new mail folder.

order the order number of this folder in its parent folder.

type the type of the new folder (can be M_FOLDER_MESSAGES or M_FOLDER_CONTACTS).

Returns:

0 on failure, or the numerical id of the newly inserted folder in the database on success.

Definition at line 91 of file datatier.c.

References `mm_wipe()`, `st_char_get()`, `st_length_get()`, and `stmt_insert()`.

Referenced by `contact_folder_create()`, and `message_folder_create()`.

5.184.4.17 `stringer_t* magma_folder_name (inx_t * folders, magma_folder_t * target)`

TODO: The assumption is that message folders will start using the interface as well. Get the fully expanded name of a folder.

Note:

The folder hierarchy will be delimited with a "." character.

Parameters:

folders the inx object holding all of a user's magma folders.

target a pointer to the magma folder object to have its name expanded.

Returns:

NULL on failure or a pointer to a managed string with the fully expanded name of the specified folder.

Definition at line 19 of file folders.c.

References FOLDER_RECURSION_LIMIT, HEAP, JOINTED, log_pedantic, magma_folder_find_number(), MANAGED_T, PLACER, st_append, st_dupe_opts(), and st_empty.

Referenced by imap_list(), magma_folder_find_full_name(), and portal_endpoint_folders_rename().

5.184.4.18 bool_t magma_folder_rename (uint64_t usernum, uint64_t foldernum, uint_t type, stringer_t * rename)

Rename a folder in the database.

Parameters:

usernum the numerical id of the user requesting the folder renaming.

foldernum the numerical id of the folder to be renamed.

type the type of the folder to be renamed (can be M_FOLDER_MESSAGES or M_FOLDER_CONTACTS).

rename a managed string containing the new name of the specified folder.

Returns:

true on success or false on failure.

Definition at line 176 of file datatier.c.

References mm_wipe(), st_char_get(), st_length_get(), and stmt_exec_affected().

Referenced by contact_folder_rename().

5.184.4.19 message_folder_t* message_folder_alloc (uint64_t foldernum, uint64_t parent, uint32_t order, stringer_t * name)

messages.c messages.c

Parameters:

foldernum the numerical id of the new message folder.

parent the numerical id of the parent folder containing the target message folder.

order the order number of this folder in its parent folder.

name a managed string containing the name of the new message folder.

Returns:

NULL on failure or a pointer to the newly allocated and initialized message folder object on success.

Definition at line 28 of file messages.c.

References inx_alloc(), M_INX_TREE, magma_folder_alloc(), message_free(), and mm_cleanup.

Referenced by magma_folder_funcs(), and message_folder_create().

5.184.4.20 `int_t message_folder_create (uint64_t usernum, uint64_t parent, stringer_t * name, inx_t * folders)`

TODO: Add validation: check whether the parent exists, and enforce the length and depth limits. Also make sure the name doesn't conflict with any built in folder names. Finally, setup a list of error codes and return the appropriate value.

Note:

This function isn't called from anywhere else at the moment.

Definition at line 45 of file messages.c.

References `inx_insert()`, `log_pedantic`, `M_FOLDER_MESSAGES`, `M_TYPE_UINT64`, `magma_folder_children()`, `magma_folder_insert()`, `message_folder_alloc()`, `message_folder_free()`, `st_empty`, `multi_t::u64`, and `multi_t::val`.

5.184.4.21 `void message_folder_free (message_folder_t * folder)`

Free a message folder object.

Returns:

This function returns no value.

Definition at line 14 of file messages.c.

References `magma_folder_free()`.

Referenced by `magma_folder_funcs()`, and `message_folder_create()`.

5.184.4.22 `bool_t message_folder_remove (uint64_t usernum, uint64_t foldernum, inx_t * folders)`

Remove a user's message folder.

See also:

[magma_folder_delete\(\)](#)

Note:

This operation will only succeed if the specified message folder is empty.

Parameters:

usernum the numerical id of the user requesting the message folder removal.

foldernum the numerical id of the folder to be removed.

folders an inx holder containing the message folders to be searched for the folder specified for removal.

Returns:

true on success or false on failure.

Definition at line 81 of file messages.c.

References `inx_count()`, `inx_delete()`, `inx_find()`, `log_pedantic`, `M_FOLDER_MESSAGES`, `M_TYPE_UINT64`, `magma_folder_children()`, and `magma_folder_delete()`.

5.184.5 Variable Documentation

5.184.5.1 `magma_folder_t`

Definition at line 28 of file folders.h.

Referenced by `contact_folder_create()`, `magma_folder_alloc()`, `magma_folder_children()`, `magma_folder_fetch()`, `magma_folder_find_full_name()`, `magma_folder_find_name()`, `portal_endpoint_folders_list()`, and `portal_endpoint_folders_rename()`.

5.185 src/objects/folders/messages.c File Reference

```
#include "magma.h"
```

Functions

- void `message_folder_free` (`message_folder_t` *folder)
Free a message folder object.
- `message_folder_t` * `message_folder_alloc` (`uint64_t` foldernum, `uint64_t` parent, `uint32_t` order, `stringer_t` *name)
Create and initialize a new message folder object.
- `int_t` `message_folder_create` (`uint64_t` usernum, `uint64_t` parent, `stringer_t` *name, `inx_t` *folders)
- `bool_t` `message_folder_remove` (`uint64_t` usernum, `uint64_t` foldernum, `inx_t` *folders)
Remove a user's message folder.

5.185.1 Function Documentation

5.185.1.1 `message_folder_t`* `message_folder_alloc` (`uint64_t` foldernum, `uint64_t` parent, `uint32_t` order, `stringer_t` * name)

Create and initialize a new message folder object. messages.c

Parameters:

- foldernum* the numerical id of the new message folder.
- parent* the numerical id of the parent folder containing the target message folder.
- order* the order number of this folder in its parent folder.
- name* a managed string containing the name of the new message folder.

Returns:

NULL on failure or a pointer to the newly allocated and initialized message folder object on success.

Definition at line 28 of file messages.c.

References `inx_alloc()`, `M_INX_TREE`, `magma_folder_alloc()`, `message_free()`, and `mm_cleanup`.

Referenced by `magma_folder_funcs()`, and `message_folder_create()`.

5.185.1.2 `int_t` `message_folder_create` (`uint64_t` usernum, `uint64_t` parent, `stringer_t` * name, `inx_t` * folders)

TODO: Add validation: check whether the parent exists, and enforce the length and depth limits. Also make sure the name doesn't conflict with any built in folder names. Finally, setup a list of error codes and return the appropriate value.

Note:

This function isn't called from anywhere else at the moment.

Definition at line 45 of file messages.c.

References `inx_insert()`, `log_pedantic`, `M_FOLDER_MESSAGES`, `M_TYPE_UINT64`, `magma_folder_children()`, `magma_folder_insert()`, `message_folder_alloc()`, `message_folder_free()`, `st_empty`, `multi_t::u64`, and `multi_t::val`.

5.185.1.3 void message_folder_free (message_folder_t **folder*)

Free a message folder object.

Returns:

This function returns no value.

Definition at line 14 of file messages.c.

References magma_folder_free().

Referenced by magma_folder_funcs(), and message_folder_create().

5.185.1.4 bool_t message_folder_remove (uint64_t *usernum*, uint64_t *foldernum*, inx_t **folders*)

Remove a user's message folder.

See also:

[magma_folder_delete\(\)](#)

Note:

This operation will only succeed if the specified message folder is empty.

Parameters:

usernum the numerical id of the user requesting the message folder removal.

foldernum the numerical id of the folder to be removed.

folders an inx holder containing the message folders to be searched for the folder specified for removal.

Returns:

true on success or false on failure.

Definition at line 81 of file messages.c.

References inx_count(), inx_delete(), inx_find(), log_pedantic, M_FOLDER_MESSAGES, M_TYPE_UINT64, magma_folder_children(), and magma_folder_delete().

5.186 src/objects/messages/messages.c File Reference

```
#include "magma.h"
```

Functions

- void `message_free` (`message_t` *message)
Free a message object and its underlying data.
- `message_t` * `message_alloc` (uint64_t messagenum, uint64_t created, uint64_t signature, uint64_t key, uint64_t flags, `stringer_t` *server, size_t size)
Allocate a new message object.
- `inx_t` * `messages_update` (uint64_t usernum)
Fetch all of a user's message folders and their child messages from the database.

5.186.1 Function Documentation

5.186.1.1 `message_t`* `message_alloc` (uint64_t *messagenum*, uint64_t *created*, uint64_t *signature*, uint64_t *key*, uint64_t *flags*, `stringer_t` * *server*, size_t *size*)

Allocate a new message object. messages.c

Parameters:

messagenum the numerical message id of the new message.
created the message creation timestamp.
signature the message's spam filter signature number.
key the message's spam filter access key.
flags the message's flags value.
server a managed string containing the name of the server where the message will be stored.
size the size, in bytes, of the raw message data.

Returns:

NULL on failure, or the newly allocated message object on success.

Definition at line 60 of file messages.c.

References `align()`, `FOREIGNDATA`, `JOINTED`, `log_pedantic`, `message_t`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `PLACER_T`, `placer_t`, `rwlock_init()`, `st_data_get()`, `st_length_get()`, and `STACK`.

Referenced by `meta_data_fetch_folder_messages()`.

5.186.1.2 void `message_free` (`message_t` * *message*)

Free a message object and its underlying data.

Returns:

This function returns no value.

Definition at line 39 of file messages.c.

References `mm_cleanup`, `rwlock_destroy()`, and `st_cleanup`.

Referenced by `message_folder_alloc()`, and `meta_data_fetch_folder_messages()`.

5.186.1.3 `inx_t* messages_update(uint64_t usernum)`

Fetch all of a user's message folders and their child messages from the database.

Parameters:

usernum the numerical id of the target user.

Returns:

NULL on failure or an `inx` object holding all the retrieved and populated message folder objects on success.

Definition at line 96 of file `messages.c`.

References `inx_cleanup()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `log_pedantic`, `M_FOLDER_MESSAGES`, `magma_folder_fetch()`, and `meta_data_fetch_folder_messages()`.

Referenced by `meta_update_message_folders()`.

5.187 src/providers/prime/messages/messages.c File Reference

```
#include "magma.h"
```

Functions

- void [encrypted_message_free](#) ([prime_message_t](#) *object)
 - void [encrypted_message_cleanup](#) ([prime_message_t](#) *object)
 - [prime_message_t](#) * [encrypted_message_alloc](#) (void)
- messages.c*
- [stringer_t](#) * [naked_message_get](#) ([stringer_t](#) *message, [prime_org_signet_t](#) *org, [prime_user_key_t](#) *user)
 - [prime_message_t](#) * [naked_message_set](#) ([stringer_t](#) *message, [prime_org_key_t](#) *destination, [prime_user_signet_t](#) *recipient)

5.187.1 Function Documentation

5.187.1.1 [prime_message_t](#)* [encrypted_message_alloc](#) (void)

messages.c

Definition at line 57 of file *messages.c*.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_message_t](#).

Referenced by [naked_message_set\(\)](#).

5.187.1.2 void [encrypted_message_cleanup](#) ([prime_message_t](#) * *object*)

Definition at line 50 of file *messages.c*.

References [encrypted_message_free\(\)](#).

5.187.1.3 void [encrypted_message_free](#) ([prime_message_t](#) * *object*)

Definition at line 10 of file *messages.c*.

References [ed25519_free\(\)](#), [encrypted_chunk_free\(\)](#), [ephemeral_chunk_free\(\)](#), [log_pedantic](#), [mm_free\(\)](#), [secp256k1_free\(\)](#), and [st_free\(\)](#).

Referenced by [encrypted_message_cleanup\(\)](#), [naked_message_set\(\)](#), and [prime_free\(\)](#).

5.187.1.4 [stringer_t](#)* [naked_message_get](#) ([stringer_t](#) * *message*, [prime_org_signet_t](#) * *org*, [prime_user_key_t](#) * *user*)

Definition at line 73 of file *messages.c*.

References [chunk_header_read\(\)](#), [data](#), [encrypted_chunk_get\(\)](#), [ephemeral_chunk_cleanup\(\)](#), [ephemeral_chunk_free\(\)](#), [ephemeral_chunk_set\(\)](#), [keks_free\(\)](#), [keks_set\(\)](#), [mm_wipe\(\)](#), [part_decrypt\(\)](#), [pl_init\(\)](#), [pl_null\(\)](#), [PLACER](#), [placer_t](#), [PRIME_CHUNK_COMMON](#), [PRIME_CHUNK_EPHEMERAL](#), [PRIME_CHUNK_HEADERS](#), [prime_chunk_keks_t](#), [prime_chunk_keys_t](#), [prime_ephemeral_chunk_t](#), [prime_header_read\(\)](#), [PRIME_MESSAGE_NAKED](#), [PRIME_SIGNATURE_DESTINATION](#), [PRIME_SIGNATURE_TREE](#), [prime_signature_tree_t](#), [PRIME_SIGNATURE_USER](#), [signature_full_verify\(\)](#), [signature_tree_add\(\)](#), [signature_tree_alloc\(\)](#), [signature_tree_free\(\)](#), [signature_tree_verify\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [st_merge](#), and [type\(\)](#).

Referenced by [prime_message_decrypt\(\)](#).

5.187.1.5 `prime_message_t*` `naked_message_set` (`stringer_t *` *message*, `prime_org_key_t *` *destination*, `prime_user_signet_t *` *recipient*)

LOW: Effective, albeit kludgy, logic to ensure common headers are formatted correctly, and each header field is reformatted values reside on a single line.

Definition at line 228 of file `messages.c`.

References `common`, `ed25519_generate()`, `encrypted_chunk_buffer()`, `encrypted_chunk_set()`, `encrypted_message_alloc()`, `encrypted_message_free()`, `ephemeral_chunk_buffer()`, `ephemeral_chunk_get()`, `HEAP`, `JOINTED`, `keys_get()`, `length`, `mail_header_end()`, `mail_header_fetch_cleaned()`, `MANAGED_T`, `MANAGEDBUF`, `mm_copy()`, `part_buffer()`, `part_encrypt()`, `pl_init()`, `pl_null()`, `PLACER`, `placer_t`, `PRIME_CHUNK_BODY`, `PRIME_CHUNK_COMMON`, `PRIME_CHUNK_FLAG_NONE`, `PRIME_CHUNK_HEADERS`, `prime_encrypted_chunk_t`, `PRIME_MESSAGE_NAKED`, `prime_message_t`, `PRIME_SIGNATURE_DESTINATION`, `prime_signature_tree_t`, `PRIME_SIGNATURE_USER`, `secp256k1_generate()`, `secp256k1_public_get()`, `secp256k1_public_set()`, `signature_full_get()`, `signature_tree_add()`, `signature_tree_alloc()`, `signature_tree_free()`, `signature_tree_get()`, `st_alloc_opts()`, `st_append`, `st_cleanup`, `st_data_get()`, `st_free()`, `st_length_get()`, `st_merge`, `st_write`, and `type()`.

Referenced by `prime_message_encrypt()`.

5.188 src/servers/imap/messages.c File Reference

```
#include "magma.h"
```

Functions

- [int_t imap_append_message](#) ([connection_t](#) *con, [meta_folder_t](#) *folder, uint32_t flags, [stringer_t](#) *message, uint64_t *outnum)
messages.c
- [int_t imap_message_expunge](#) ([connection_t](#) *con, [meta_message_t](#) *message)
- [int_t imap_message_copier](#) ([connection_t](#) *con, [meta_message_t](#) *message, uint64_t target, uint64_t *outnum)

5.188.1 Function Documentation

5.188.1.1 [int_t imap_append_message](#) ([connection_t](#) *con, [meta_folder_t](#) *folder, uint32_t flags, [stringer_t](#) *message, uint64_t *outnum)

messages.c

Definition at line 10 of file messages.c.

References [magma_t::active](#), [inx_append\(\)](#), [log_pedantic](#), [M_TYPE_UINT64](#), [magma](#), [MAIL_STATUS_APPENDED](#), [MAIL_STATUS_RECENT](#), [mail_store_message\(\)](#), [meta_message_t](#), [meta_messages_update_sequences\(\)](#), [META_USER_ENCRYPT_DATA](#), [mm_alloc\(\)](#), [mm_free\(\)](#), [OBJECT_MESSAGES](#), [serial_get\(\)](#), [serial_increment\(\)](#), [st_char_get\(\)](#), [st_length_get\(\)](#), [st_length_int\(\)](#), [magma_t::storage](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [imap_append\(\)](#).

5.188.1.2 [int_t imap_message_copier](#) ([connection_t](#) *con, [meta_message_t](#) *message, uint64_t target, uint64_t *outnum)

Definition at line 65 of file messages.c.

References [inx_append\(\)](#), [log_pedantic](#), [M_TYPE_UINT64](#), [mail_copy_message\(\)](#), [MAIL_STATUS_DELETED](#), [MAIL_STATUS_HIDDEN](#), [MAIL_STATUS_RECENT](#), [meta_message_dupe\(\)](#), [meta_message_t](#), [meta_messages_update_sequences\(\)](#), [mm_free\(\)](#), [OBJECT_MESSAGES](#), [serial_get\(\)](#), [serial_increment\(\)](#), [status](#), [multi_t::type](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [imap_copy\(\)](#).

5.188.1.3 [int_t imap_message_expunge](#) ([connection_t](#) *con, [meta_message_t](#) *message)

Definition at line 53 of file messages.c.

References [inx_delete\(\)](#), [M_TYPE_UINT64](#), and [mail_remove_message\(\)](#).

Referenced by [imap_close\(\)](#), and [imap_expunge\(\)](#).

5.189 src/web/portal/messages.c File Reference

```
#include "magma.h"
```

Functions

- json_t * [portal_message_meta](#) (meta_message_t *meta)
- json_t * [portal_message_source](#) (meta_message_t *meta)
- json_t * [portal_message_security](#) (meta_message_t *meta)
- json_t * [portal_message_server](#) (meta_message_t *meta)
- json_t * [portal_message_header](#) (meta_message_t *meta, mail_message_t *data)
Build the "header" section response to a rpc-json "messages.load" request.
- json_t * [portal_message_body](#) (meta_message_t *meta, mail_message_t *data)
- json_t * [portal_message_attachments](#) (meta_message_t *meta, mail_message_t *data)
messages.c
- json_t * [portal_message_info](#) (meta_message_t *meta)
Build the "info" section response to a rpc-json "messages.load" request.

5.189.1 Function Documentation

5.189.1.1 json_t* portal_message_attachments (meta_message_t * meta, mail_message_t * data)

messages.c

LOW: We also need to detect messages that have no readable content so the entire body is just a blob.

Create a function to search for the filename. Common locations are: Content-Type: image/png; name="webmail-php-download.png" Content-Disposition: attachment; filename="webmail-php-download.png"

Definition at line 164 of file messages.c.

References [ar_field_ptr\(\)](#), [ar_length_get\(\)](#), [mail_mime_t::body](#), [mail_mime_t::children](#), [count](#), [mail_mime_t::header](#), [json_array_append_new_d](#), [json_array_d](#), [json_decref_d](#), [json_pack_ex_d](#), [log_pedantic](#), [mail_mime_type_group\(\)](#), [mail_mime_type_sub\(\)](#), [MESSAGE_TYPE_HTML](#), [MESSAGE_TYPE_MULTI_MIXED](#), [MESSAGE_TYPE_PLAIN](#), [mail_message_t::mime](#), [PLACER](#), [st_aprint\(\)](#), [st_char_get\(\)](#), [st_cleanup](#), [st_length_get\(\)](#), [st_merge](#), [mail_mime_t::type](#), and [type\(\)](#).

Referenced by [portal_endpoint_messages_load\(\)](#).

5.189.1.2 json_t* portal_message_body (meta_message_t * meta, mail_message_t * data)

TODO: I consider it a miracle whenever the above logic actually manages to select the message content. But if it doesn't we fall through to this fixed error message, at least until we can improve the selection process.

HIGH: Because JSON wants NULLERS, and were using PLACEHOLDERS, its printing out the entire message, not just the section of interest.

Definition at line 108 of file messages.c.

References [ar_field_ptr\(\)](#), [mail_mime_t::body](#), [mail_mime_t::children](#), [CONTIGUOUS](#), [HEAP](#), [json_pack_ex_d](#), [log_pedantic](#), [MAIL_MIME_RECURSION_LIMIT](#), [MESSAGE_TYPE_HTML](#), [MESSAGE_TYPE_MULTI_ALTERNATIVE](#), [MESSAGE_TYPE_MULTI_MIXED](#), [MESSAGE_TYPE_MULTI_RELATED](#), [MESSAGE_TYPE_MULTI_UNKOWN](#), [MESSAGE_TYPE_PLAIN](#), [mail_message_t::mime](#), [NULLER_T](#), [pl_char_get\(\)](#), [pl_length_get\(\)](#), [PLACER](#), [st_char_get\(\)](#), [st_cleanup](#), [st_dupe_opts\(\)](#), [st_nullify\(\)](#), [mail_mime_t::type](#), and [type\(\)](#).

Referenced by [portal_endpoint_messages_load\(\)](#).

5.189.1.3 json_t* portal_message_header (meta_message_t * meta, mail_message_t * data)

Build the "header" section response to a rpc-json "messages.load" request.

Note:

The following header fields will be returned: To, CC, BCC, From, Replyto, Sender, Return-Path, Subject, Date, and Size.

Parameters:

meta a pointer to the meta message object of the requested message.

data a pointer to the mail message object containing the requested message's data.

Returns:

a pointer to a json object containing the header fields of the requested message.

Definition at line 79 of file messages.c.

References mail_message_t::header_length, json_pack_ex_d, log_pedantic, mail_header_fetch_cleaned(), mm_wipe(), PLACER, st_char_get(), st_cleanup, and mail_message_t::text.

Referenced by portal_endpoint_messages_load().

5.189.1.4 json_t* portal_message_info (meta_message_t * meta)

Build the "info" section response to a rpc-json "messages.load" request.

Parameters:

meta a pointer to the meta message object of the requested message.

Returns:

a pointer to a json object containing the appropriate information about the requested message.

Definition at line 270 of file messages.c.

References json_pack_ex_d, and log_pedantic.

Referenced by portal_endpoint_messages_load().

5.189.1.5 json_t* portal_message_meta (meta_message_t * meta)

Definition at line 13 of file messages.c.

References json_pack_ex_d, log_pedantic, portal_message_flags_array(), and portal_message_tags_array().

Referenced by portal_endpoint_messages_load().

5.189.1.6 json_t* portal_message_security (meta_message_t * meta)

TODO: Replace hard coded values with actual data.

Definition at line 42 of file messages.c.

References json_pack_ex_d, and log_pedantic.

Referenced by portal_endpoint_messages_load().

5.189.1.7 json_t* portal_message_server (meta_message_t * meta)

TODO: Replace hard coded values with actual data.

Definition at line 56 of file messages.c.

References json_pack_ex_d, and log_pedantic.

Referenced by portal_endpoint_messages_load().

5.189.1.8 json_t* portal_message_source (meta_message_t * meta)

TODO: Replace hard coded values with actual data.

Definition at line 27 of file messages.c.

References json_pack_ex_d, log_pedantic, and rand_get_int64().

Referenced by portal_endpoint_messages_load().

5.190 src/objects/locks.c File Reference

```
#include "magma.h"
```

Defines

- #define [MAGMA_LOCK_TIMEOUT](#) 60
- #define [MAGMA_LOCK_EXPIRATION](#) 600

Functions

- [int_t lock_get](#) ([stringer_t](#) *key)
Acquire a named lock, with synchronization provided via memcached.
- void [lock_release](#) ([stringer_t](#) *key)
Release a named lock, with synchronization provided via memcached.
- [int_t user_lock](#) (uint64_t usernum)
Acquire a lock in the magma.user keyspace.
- void [user_unlock](#) (uint64_t usernum)
Unlock a lock in the magma.user keyspace.

5.190.1 Define Documentation

5.190.1.1 #define MAGMA_LOCK_EXPIRATION 600

Definition at line 11 of file locks.c.

Referenced by [lock_get\(\)](#).

5.190.1.2 #define MAGMA_LOCK_TIMEOUT 60

Definition at line 10 of file locks.c.

Referenced by [lock_get\(\)](#).

5.190.2 Function Documentation

5.190.2.1 int_t lock_get (stringer_t * key)

Acquire a named lock, with synchronization provided via memcached. [locks.c](#)

See also:

[cache_silent_add\(\)](#)

Note:

The lock will be held for a maximum of 10 minutes, and failed locking attempts will be retried periodically for a maximum of 1 minute before returning failure.

Parameters:

key a managed string containing the name of the lock to be acquired.

Returns:

-1 on general failure, 0 on memcached failure, or 1 on success.

Definition at line 21 of file locks.c.

References `cache_silent_add()`, `lock`, `log_pedantic`, `MAGMA_LOCK_EXPIRATION`, `MAGMA_LOCK_TIMEOUT`, `MANAGEDBUF`, `PLACER`, `st_char_get()`, `st_empty`, `st_length_int()`, and `st_sprint()`.

Referenced by `api_endpoint_auth()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_reply()`, and `user_lock()`.

5.190.2.2 void lock_release (stringer_t * *key*)

Release a named lock, with synchronization provided via memcached.

See also:

[cache_delete\(\)](#)

Note:

The lock will be held for 10 seconds, and locking attempts will occur periodically for 60 seconds prior to failure.

Parameters:

key a managed string containing the name of the lock to be released.

Returns:

-1 on general failure, 0 on memcached failure, or 1 on success.

LOW: At some point we should add logic to check whether this cluster node even owns the lock before blindly deleting the lock.

Definition at line 61 of file locks.c.

References `cache_delete()`, `lock`, `log_pedantic`, `MANAGEDBUF`, `st_char_get()`, `st_empty`, `st_length_int()`, and `st_sprint()`.

Referenced by `api_endpoint_auth()`, `imap_login()`, `pop_pass()`, `portal_endpoint_auth()`, `smtp_auth_login()`, `smtp_auth_plain()`, `smtp_reply()`, and `user_unlock()`.

5.190.2.3 int_t user_lock (uint64_t *usernum*)

Acquire a lock in the magma.user keyspace.

Parameters:

usernum the numerical id of the user for whom the lock will be acquired.

Returns:

-1 on general failure, 0 on memcached failure, or 1 on success.

Definition at line 82 of file locks.c.

References `lock_get()`, `MANAGEDBUF`, and `st_sprint()`.

Referenced by `imap_close()`, `imap_copy()`, `imap_expunge()`, `smtp_rollout()`, and `smtp_store_message()`.

5.190.2.4 void user_unlock (uint64_t *usernum*)

Unlock a lock in the magma.user keyspace.

Parameters:

usernum the numerical id of the user for whom the lock will be unlocked.

Returns:

This function returns no value.

Definition at line 98 of file locks.c.

References lock_release(), MANAGEDBUF, and st_sprint().

Referenced by imap_close(), imap_copy(), imap_expunge(), smtp_rollout(), and smtp_store_message().

5.191 src/objects/mail/cleanup.c File Reference

```
#include "magma.h"
```

Functions

- void [mail_destroy_header](#) ([stringer_t](#) *header)
Destroy a mail header.
- [bool_t mail_message_cleanup](#) ([stringer_t](#) **message)
Clean up the body of a message read in via smtp, enforcing magma.smtp.wrap_line_length.

5.191.1 Function Documentation

5.191.1.1 void mail_destroy_header (stringer_t * header)

Destroy a mail header. [cleanup.c](#)

Parameters:

header a managed string containing the mail header to be freed.

Returns:

This function returns nov alue.

Definition at line 15 of file cleanup.c.

References [st_cleanup](#).

Referenced by [imap_fetch_body\(\)](#), [imap_fetch_message\(\)](#), [imap_fetch_return_message\(\)](#), [imap_fetch_return_mime\(\)](#), [imap_fetch_return_text\(\)](#), and [imap_search_messages\(\)](#).

5.191.1.2 bool_t mail_message_cleanup (stringer_t ** message)

Clean up the body of a message read in via smtp, enforcing magma.smtp.wrap_line_length.

Note:

This function fixes broken line separators by making sure each is followed by and vice versa. All Return-Path: header lines are also removed. New lines are begun whenever the current length of any line reaches the configuration value set in magma.smtp.wrap_line_length. The trailing dot at the end of the smtp DATA command is also stripped. If the original message ends with , it will have appended to it.

Parameters:

message a pointer to a managed string that contains the message input, and will also store the cleaned output on success.

Returns:

true on success or false on failure.

LOW: Splitting lines, particularly headers causes more problems then it fixes, so this code is currently disabled. At some point we should survey other mail systems and see how they handle long lines.

Definition at line 31 of file cleanup.c.

References HEAP, JOINTED, length, log_pedantic, magma, MAPPED_T, mm_cmp_ci_eq(), magma_t::smtp, st_alloc_opts(), st_char_get(), st_free(), st_length_get(), st_length_set(), and magma_t::wrap_line_length.

Referenced by smtp_data().

5.192 src/objects/mail/counters.c File Reference

```
#include "magma.h"
```

Functions

- `uint32_t mail_count_received (stringer_t *message)`
Count the number of "Received:" lines in a mail message.
- `size_t mail_header_end (stringer_t *message)`
Get the number of bytes taken up by a mail header.

5.192.1 Function Documentation

5.192.1.1 `uint32_t mail_count_received (stringer_t * message)`

Count the number of "Received:" lines in a mail message. counters.c

Parameters:

message a managed string containing the mail message to be scanned.

Returns:

the number of matching lines found, or 0 on failure.

Definition at line 15 of file counters.c.

References `log_pedantic`, `PLACER`, `st_cmp_ci_starts()`, and `st_empty_out()`.

Referenced by `smtp_data()`.

5.192.1.2 `size_t mail_header_end (stringer_t * message)`

Get the number of bytes taken up by a mail header.

Parameters:

message a managed string containing the full smtp mail message.

Returns:

0 on failure, or the number of bytes occupied by the mail header.

Definition at line 56 of file counters.c.

References `length`, `log_pedantic`, `st_char_get()`, and `st_length_get()`.

Referenced by `mail_add_forward_headers()`, `mail_add_outbound_headers()`, `mail_add_required_headers()`, `mail_create_message()`, `mail_headers()`, `mail_message()`, `mail_mod_subject()`, `naked_message_set()`, and `smtp_check_filters()`.

5.193 src/providers/consumers/counters.c File Reference

```
#include "magma.h"
```

Functions

- int [stamp_counter_check](#) (char *key, size_t keylen, unsigned long interval)
- void [stamp_counter_increment](#) (char *key, size_t keylen)

5.193.1 Function Documentation

5.193.1.1 int stamp_counter_check (char * *key*, size_t *keylen*, unsigned long *interval*)

Definition at line 10 of file counters.c.

References [cache_get\(\)](#), [count](#), [length](#), [log_pedantic](#), [PLACER](#), [placer_t](#), [st_free\(\)](#), [tok_get_count_st\(\)](#), [tok_get_st\(\)](#), and [uint64_conv_pl\(\)](#).

5.193.1.2 void stamp_counter_increment (char * *key*, size_t *keylen*)

Definition at line 41 of file counters.c.

References [cache_append\(\)](#), [log_pedantic](#), and [PLACER](#).

5.194 src/objects/mail/headers.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * mail_header_fetch_cleaned](#) ([stringer_t](#) *header, [stringer_t](#) *key)
Fetch the value of a specified line from a mail header, performing whitespace and line cleaning.
- [stringer_t * mail_header_fetch_all](#) ([stringer_t](#) *header, [stringer_t](#) *key)
Get all the values that match a named header line in a message header.
- [placer_t mail_header_pop](#) ([stringer_t](#) *header, [size_t](#) *position)
Pop the next header line from a mail message header.
- [placer_t mail_store_header](#) ([chr_t](#) *stream, [size_t](#) length)
Return a placer pointing to the trimmed value of a mail header.
- [bool_t mail_headers](#) ([smtp_message_t](#) *message)
Parse an smtp message header and store the To, From, Date, and Subject header values.
- [void mail_mod_subject](#) ([stringer_t](#) **message, [chr_t](#) *label)
Prepend text to a Subject header line inside of a mail message.
- [bool_t mail_add_required_headers](#) ([connection_t](#) *con, [smtp_message_t](#) *message)
Generate any missing required headers for an smtp message object.
- [stringer_t * mail_add_inbound_headers](#) ([connection_t](#) *con, [smtp_inbound_prefs_t](#) *prefs)
Prepend the Return-Path: and Received: headers to an inbound smtp message.
- [void mail_add_forward_headers](#) ([server_t](#) *server, [stringer_t](#) **message, [stringer_t](#) *id, [int_t](#) mark, [uint64_t](#) signum, [uint64_t](#) sigkey)
Add necessary headers to a forwarded mail message.
- [int_t mail_add_outbound_headers](#) ([connection_t](#) *con)
Add a Received: header and dkim signature to an outbound relayed smtp message.

5.194.1 Function Documentation

5.194.1.1 void mail_add_forward_headers (server_t * server, stringer_t ** message, stringer_t * id, int_t mark, uint64_t signum, uint64_t sigkey)

Add necessary headers to a forwarded mail message. headers.c

Note:

This function will prepend tags to the Subject line like "JUNK", "INFECTED", "SPOOFED", etc. The following headers will be stripped from the message: Sender, Return-Path, DKIM-Signature, and DomainKey-Signature. This daemon will be reinserted into the headers as the origin of the Sender: header value. If signum and sigkey are both set, then a spam training url will also be inserted into the message. Finally, if [magma.dkim.enabled](#) is set, a dkim signature will be added to the message.

Parameters:

server the server object of the web server where the teacher application is hosted.

message the address of a managed string containing the message data that will be set to the modified messagee data on success.

id a managed string containing the message id (for dkim).

mark a bitmask of marks to be tagged in the message subject (SMTP_MARK_SPAM, SMTP_MARK_VIRUS, SMTP_MARK_SPOOF, SMTP_MARK_RBL, and SMTP_MARK_PHISH).

signum the optional spam signature number referenced by the teacher url.

sigkey the optional spam signature key for client verification in the teacher app.

Returns:

This function returns no value. LOW: We should be passing the message as a `stringer_t *` and then returning it as a `stringer_t *`.

Definition at line 730 of file headers.c.

References `CONSTANT`, `magma_t::dkim`, `dkim_signature_create()`, `magma_t::enabled`, `HEAP`, `JOINTED`, `log_pedantic`, `magma`, `mail_destroy()`, `mail_header_end()`, `mail_header_pop()`, `MAIL_MARK_JUNK`, `mail_message()`, `mail_mod_subject()`, `mail_signature_add()`, `MAPPED_T`, `mm_cmp_ci_eq()`, `pl_empty()`, `PLACER`, `placer_t`, `SMTP_MARK_PHISH`, `SMTP_MARK_RBL`, `SMTP_MARK_SPAM`, `SMTP_MARK_SPOOF`, `SMTP_MARK_VIRUS`, `st_alloc_opts()`, `st_append_opts()`, `st_char_get()`, `st_cleanup`, `st_cmp_ci_starts()`, `st_cmp_cs_starts()`, `st_data_get()`, `st_dupe()`, `st_dupe_opts()`, `st_free()`, `st_length_get()`, `st_merge_opts()`, `status`, and `mail_message_t::text`.

Referenced by `smtp_forward_message()`.

5.194.1.2 `stringer_t* mail_add_inbound_headers(connection_t * con, smtp_inbound_prefs_t * prefs)`

Prepend the Return-Path: and Received: headers to an inbound smtp message.

Parameters:

con the connection across which the inbound smtp message was accepted.

prefs the smtp inbound preferences corresponding to the connection, with the rcptto field populated.

Returns:

NULL on failure, or a managed string containing the message data preceded by the Return-Path and Received headers on success.

Definition at line 638 of file headers.c.

References `con_addr_presentation()`, `con_reverse_check()`, `CONSTANT`, `HEAP`, `JOINTED`, `log_pedantic`, `MANAGEDBUF`, `MAPPED_T`, `smtp_inbound_prefs_t::rcptto`, `st_cmp_ci_eq()`, `st_cmp_cs_eq()`, and `st_merge_opts()`.

Referenced by `smtp_accept_message()`.

5.194.1.3 `int_t mail_add_outbound_headers(connection_t * con)`

Add a Received: header and dkim signature to an outbound relayed smtp message.

Note:

If the message already has a Received header, the dkim signature will be inserted right before the first instance.

Parameters:

con the connection across which the outbound smtp message is being sent.

Returns:

NULL on failure, or a managed string containing the message data preceded by the Received header and including the dkim signature on success.

Definition at line 849 of file headers.c.

References `con_addr_presentation()`, `con_reverse_check()`, `magma_t::dkim`, `dkim_signature_create()`, `domain_dkim()`, `magma_t::enabled`, `HEAP`, `JOINTED`, `log_pedantic`, `magma`, `mail_domain_get()`, `mail_header_end()`, `mail_header_pop()`, `MANAGEDBUF`, `MAPPED_T`, `mm_cmp_ci_eq()`, `pl_empty()`, `PLACER`, `placer_t`, `st_char_get()`, `st_cleanup`, `st_cmp_cs_eq()`, `st_data_get()`, `st_dupe()`, `st_empty`, `st_free()`, `st_import()`, `st_length_get()`, and `st_merge_opts()`.

Referenced by `smtp_relay_message()`.

5.194.1.4 `bool_t mail_add_required_headers (connection_t * con, smtp_message_t * message)`

Generate any missing required headers for an smtp message object.

Note:

These required headers include To, From, Subject and Date, and if they are absent, they will be generated from the specified connection's inbound or outbound preferences, accordingly.

Parameters:

con a pointer to the connection object across which the smtp message was received.

message the smtp message object to be updated for any missing required headers.

Returns:

true on success or false on failure.

Definition at line 457 of file headers.c.

References `smtp_recipients_t::address`, `smtp_message_t::date`, `smtp_message_t::from`, `smtp_message_t::header_length`, `HEAP`, `JOINTED`, `length`, `log_pedantic`, `mail_header_end()`, `mail_headers()`, `MAPPED_T`, `smtp_recipients_t::next`, `smtp_inbound_prefs_t::next`, `PLACER`, `smtp_inbound_prefs_t::rcptto`, `st_alloc()`, `st_avail_get()`, `st_char_get()`, `st_cleanup`, `st_empty`, `st_free()`, `st_import()`, `st_length_get()`, `st_length_set()`, `st_merge`, `st_merge_opts()`, `smtp_message_t::subject`, `smtp_message_t::text`, and `smtp_message_t::to`.

Referenced by `smtp_data()`.

5.194.1.5 `stringer_t* mail_header_fetch_all (stringer_t * header, stringer_t * key)`

Get all the values that match a named header line in a message header.

Parameters:

header a managed string containing the message header.

key a managed string containing the header name to be searched for.

Returns:

NULL on failure or a managed string containing all the matching header line values separated by newlines.

Definition at line 147 of file headers.c.

References `mm_cmp_ci_eq()`, `PLACER`, `st_char_get()`, `st_empty`, `st_free()`, `st_import()`, `st_length_get()`, and `st_merge`.

Referenced by `imap_fetch_body_mime()`, `imap_search_messages_header()`, `mail_discover_encoding()`, `mail_discover_type()`, `mail_get_boundary()`, `mail_mime_boundary()`, and `smtp_check_filters()`.

5.194.1.6 `stringer_t* mail_header_fetch_cleaned (stringer_t * header, stringer_t * key)`

Fetch the value of a specified line from a mail header, performing whitespace and line cleaning.

Note:

This function handles mail header values that span new lines, although the output excludes line breaks and consecutive space characters.

Parameters:

header a managed string containing the full header of the target mail message.

key a managed string containing the name of the desired header name to be extracted, excluding the trailing ":"

Returns:

NULL on failure, or a managed string containing the cleaned value of the specified mail header line on success.

Definition at line 17 of file headers.c.

References PLACER, st_alloc(), st_char_get(), st_cmp_ci_starts(), st_empty, st_length_get(), st_length_set(), and st_trim().

Referenced by imap_fetch_bodystructure(), imap_fetch_envelope(), imap_search_messages_date(), mail_headers(), mail_message(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_encoding(), mail_mime_type(), mail_mime_type_group(), mail_mime_type_parameters(), mail_mime_type_sub(), naked_message_set(), portal_endpoint_messages_list(), and portal_message_header().

5.194.1.7 placer_t mail_header_pop (stringer_t * *header*, size_t * *position*)

Pop the next header line from a mail message header.

Parameters:

header a managed string containing the entire mail message header.

position a pointer to a size_t that contains the tracker index into the entire header for the pop operation, and will also receive the new value of the tracker for the subsequent call, when a new header line is located.

Returns:

[pl_null\(\)](#) on error or a placer pointing to the next mail message subject line on success.

Definition at line 236 of file headers.c.

References pl_init(), pl_null(), st_char_get(), st_empty, and st_length_get().

Referenced by imap_fetch_body_header(), mail_add_forward_headers(), mail_add_outbound_headers(), and mail_mod_subject().

5.194.1.8 bool_t mail_headers (smtp_message_t * *message*)

Parse an smtp message header and store the To, From, Date, and Subject header values.

Note:

The found header fields will be stored inside the same smtp message object passed by the caller as input.

Parameters:

message a pointer to a partially populated smtp message object that contains the raw message data to be parsed.

Returns:

true on success or false on failure.

Definition at line 318 of file headers.c.

References smtp_message_t::date, smtp_message_t::from, smtp_message_t::header_length, length, log_pedantic, mail_header_end(), mail_header_fetch_cleaned(), mail_store_header(), mm_cmp_ci_eq(), PLACER, st_char_get(), st_free(), st_length_get(), smtp_message_t::subject, smtp_message_t::text, and smtp_message_t::to.

Referenced by mail_add_required_headers(), and mail_create_message().

5.194.1.9 void mail_mod_subject (stringer_t ** *message*, chr_t * *label*)

Prepend text to a Subject header line inside of a mail message.

Note:

If no Subject line is found in the header of the message, one will be created and inserted with the specified label at the end of the header.

Parameters:

message a pointer to the address of a managed string that contains the mail message header or mail message body, and will receive the value of the resulting managed string that will be allocated to hold the modified contents.

label a null-terminated string that will be prepended to the value of the subject header value.

Returns:

This function returns no value.

Definition at line 389 of file headers.c.

References CONSTANT, log_pedantic, mail_header_end(), mail_header_pop(), pl_empty(), PLACER, placer_t, st_char_get(), st_cmp_ci_starts(), st_data_get(), st_free(), st_length_get(), and st_merge.

Referenced by mail_add_forward_headers(), mail_load_message(), and smtp_check_filters().

5.194.1.10 placer_t mail_store_header (chr_t * *stream*, size_t *length*)

Return a placer pointing to the trimmed value of a mail header.

Note:

This function merely marks a string residing between leading whitespace and a trailing or
.

Parameters:

stream a pointer to a buffer containing the start of the mail header data.

length the length, in bytes, of the mail header field. a placer containing the trimmed value of the mail header line.

Definition at line 291 of file headers.c.

References pl_init().

Referenced by mail_headers(), and mail_message().

5.195 src/providers/prime/transposition/binary/headers.c File Reference

```
#include "magma.h"
```

Functions

- `size_t prime_header_length (uint16_t type)`
Determine whether a given type uses a 5 or 6 byte header.
 - `int_t prime_header_read (stringer_t *object, uint16_t *type, uint32_t *size)`
Read the first few bytes of a serialized PRIME object and return the magic number and object size in native form.
 - `stringer_t * prime_header_write (uint16_t type, size_t size, stringer_t *output)`
 - `stringer_t * prime_header_org_signet_write (size_t size, stringer_t *output)`
 - `stringer_t * prime_header_user_signet_write (size_t size, stringer_t *output)`
 - `stringer_t * prime_header_user_signing_request_write (size_t size, stringer_t *output)`
 - `stringer_t * prime_header_org_key_write (size_t size, stringer_t *output)`
 - `stringer_t * prime_header_user_key_write (size_t size, stringer_t *output)`
 - `stringer_t * prime_header_encrypted_org_key_write (size_t size, stringer_t *output)`
 - `stringer_t * prime_header_encrypted_user_key_write (size_t size, stringer_t *output)`
 - `stringer_t * prime_header_encrypted_message_write (size_t size, stringer_t *output)`
- headers.c*

5.195.1 Function Documentation

5.195.1.1 `stringer_t* prime_header_encrypted_message_write (size_t size, stringer_t * output)`

headers.c

Definition at line 221 of file headers.c.

References `prime_header_write()`, and `PRIME_MESSAGE_ENCRYPTED`.

5.195.1.2 `stringer_t* prime_header_encrypted_org_key_write (size_t size, stringer_t * output)`

Definition at line 213 of file headers.c.

References `prime_header_write()`, and `PRIME_ORG_KEY_ENCRYPTED`.

5.195.1.3 `stringer_t* prime_header_encrypted_user_key_write (size_t size, stringer_t * output)`

Definition at line 217 of file headers.c.

References `prime_header_write()`, and `PRIME_USER_KEY_ENCRYPTED`.

5.195.1.4 `size_t prime_header_length (uint16_t type)`

Determine whether a given type uses a 5 or 6 byte header.

Parameters:

type The PRIME type.

Returns:

Returns the number of bytes used by the header, or 0 if an invalid type is supplied.

Definition at line 15 of file headers.c.

References `length`, `log_pedantic`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_header_write()`.

5.195.1.5 stringer_t* prime_header_org_key_write (size_t size, stringer_t * output)

Definition at line 205 of file headers.c.

References `prime_header_write()`, and `PRIME_ORG_KEY`.

Referenced by `org_key_get()`.

5.195.1.6 stringer_t* prime_header_org_signet_write (size_t size, stringer_t * output)

Definition at line 193 of file headers.c.

References `prime_header_write()`, and `PRIME_ORG_SIGNET`.

Referenced by `org_signet_get()`.

5.195.1.7 int_t prime_header_read (stringer_t * object, uint16_t * type, uint32_t * size)

Read the first few bytes of a serialized PRIME object and return the magic number and object size in native form.

Parameters:

data The serialized object buffer.

type A pointer where the type will be stored, if successful.

size A pointer to where the parsed size will be stored, if the read is successful.

Returns:

0 if successful, negative values if the object data is invalid, or positive numbers if a programmatic error occurs. 2 = The object buffer was empty. 1 = A null pointer was supplied. 0 = Success. -1 = The buffer is too short for a valid PRIME object header. -2 = The size in the header doesn't match the amount of data supplied. (size = header_len + object_len). -3 = The header indicates an unrecognized PRIME type.

Definition at line 53 of file headers.c.

References `log_pedantic`, `mm_copy()`, `PRIME_MESSAGE_ABUSE`, `PRIME_MESSAGE_BOUNCE`, `PRIME_MESSAGE_DRAFT`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_MESSAGE_FORWARD`, `PRIME_MESSAGE_NAKED`, `PRIME_MESSAGE_SENT`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, and `st_empty_out()`.

Referenced by `aes_artifact_encrypt()`, `naked_message_get()`, `prime_key_decrypt()`, `prime_pem_wrap()`, `prime_set()`, and `prime_unpack()`.

5.195.1.8 stringer_t* prime_header_user_key_write (size_t size, stringer_t * output)

Definition at line 209 of file headers.c.

References `prime_header_write()`, and `PRIME_USER_KEY`.

Referenced by `user_key_get()`.

5.195.1.9 stringer_t* prime_header_user_signet_write (size_t size, stringer_t * output)

Definition at line 197 of file headers.c.

References prime_header_write(), and PRIME_USER_SIGNET.

Referenced by user_signet_get().

5.195.1.10 stringer_t* prime_header_user_signing_request_write (size_t size, stringer_t * output)

Definition at line 201 of file headers.c.

References prime_header_write(), and PRIME_USER_SIGNING_REQUEST.

Referenced by user_request_get().

5.195.1.11 stringer_t* prime_header_write (uint16_t type, size_t size, stringer_t * output)**Parameters:**

type
size
output

Returns:

Definition at line 127 of file headers.c.

References length, log_error, log_pedantic, mm_copy(), prime_header_length(), PRIME_MESSAGE_ENCRYPTED, prime_object_size_max(), prime_object_size_min(), prime_object_type(), PRIME_ORG_KEY, PRIME_ORG_KEY_ENCRYPTED, PRIME_ORG_SIGNET, PRIME_USER_KEY, PRIME_USER_KEY_ENCRYPTED, PRIME_USER_SIGNET, PRIME_USER_SIGNING_REQUEST, st_data_get(), st_free(), st_length_set(), st_opt_get(), st_output(), and st_valid_tracked().

Referenced by aes_artifact_decrypt(), prime_header_encrypted_message_write(), prime_header_encrypted_org_key_write(), prime_header_encrypted_user_key_write(), prime_header_org_key_write(), prime_header_org_signet_write(), prime_header_user_key_write(), prime_header_user_signet_write(), and prime_header_user_signing_request_write().

5.196 src/objects/mail/load_message.c File Reference

```
#include "magma.h"
```

Functions

- [mail_message_t * mail_load_message](#) ([meta_message_t *meta](#), [meta_user_t *user](#), [server_t *server](#), [bool_t parse](#))
Load a stored mail message from disk.
- [stringer_t * mail_load_header](#) ([meta_message_t *meta](#), [meta_user_t *user](#), [server_t *server](#), [bool_t parse](#))
Get the header of a message, checking first in the cache and then on disk.
- [mail_message_t * mail_load_message_top](#) ([meta_message_t *meta](#), [meta_user_t *user](#), [server_t *server](#), [uint64_t lines](#), [bool_t parse](#))
Get the top of a mail message, up to a specified maximum number of lines of content.

5.196.1 Function Documentation

5.196.1.1 [stringer_t * mail_load_header](#) ([meta_message_t * meta](#), [meta_user_t * user](#), [server_t * server](#), [bool_t parse](#))

Get the header of a message, checking first in the cache and then on disk. [load_message.c](#)

Note:

The file data is first unencrypted and decompressed, according to the file header flags. When extracted, the header's Subject line is branded with any applicable labels such as JUNK, INFECTED, SPOOFED, BLACKHOLED, PHISHING.

Parameters:

meta the meta message object of the message to be queried.
user the meta user object of the user that owns the message.

Returns:

NULL on failure or a managed string containing the message's header on success.

Definition at line 237 of file [load_message.c](#).

References [mail_message_t::header_length](#), [log_pedantic](#), [mail_destroy\(\)](#), [mail_load_message\(\)](#), [st_char_get\(\)](#), [st_import\(\)](#), and [mail_message_t::text](#).

Referenced by [imap_fetch_return_header\(\)](#), [imap_search_messages_date\(\)](#), [imap_search_messages_header\(\)](#), and [portal_endpoint_messages_list\(\)](#).

5.196.1.2 [mail_message_t * mail_load_message](#) ([meta_message_t * meta](#), [meta_user_t * user](#), [server_t * server](#), [bool_t parse](#))

Load a stored mail message from disk.

Note:

The mail message will always, at the very least, be compressed using the lzo algorithm; however, on-disk encryption may be enabled. If parsing is enabled, a spam signature training link may be embedded in the message.

Parameters:

meta the meta message object of the message to be loaded from disk.

user the meta user object of the user that owns the requested message.

server the server object of the web server where the spam teacher application is hosted.

parse if true, the header's Subject line is branded with any applicable labels such as JUNK, INFECTED, SPOOFED, BLACKHOLED, PHISHING.

Returns:

NULL on failure or a a mail message object containing the retrieved mail message data on success.

Definition at line 20 of file load_message.c.

References compress_import(), decompress_lzo(), FMESSAGE_MAGIC_1, FMESSAGE_MAGIC_2, FMESSAGE_OPT_COMPRESSED, FMESSAGE_OPT_ENCRYPTED, log_info, log_pedantic, mail_cache_get(), mail_cache_set(), mail_db_hide_message(), MAIL_MARK_BLACKHOLED, MAIL_MARK_INFECTED, MAIL_MARK_JUNK, MAIL_MARK_PHISHING, MAIL_MARK_SPOOFED, mail_message(), mail_message_path(), mail_mod_subject(), mail_signature_add(), MAIL_STATUS_ENCRYPTED, message_header_t, META_USER_ENCRYPT_DATA, ns_free(), OBJECT_MESSAGES, org_signet, prime_message_decrypt(), serial_increment(), st_alloc(), st_char_get(), st_free(), st_length_int(), st_length_set(), and mail_message_t::text.

Referenced by imap_fetch_message(), imap_fetch_return_message(), imap_fetch_return_mime(), imap_fetch_return_text(), imap_search_messages_body(), imap_search_messages_text(), mail_load_header(), mail_load_message_top(), pop_retr(), and portal_endpoint_messages_load().

5.196.1.3 mail_message_t* mail_load_message_top (meta_message_t * meta, meta_user_t * user, server_t * server, uint64_t lines, bool_t parse)

Get the top of a mail message, up to a specified maximum number of lines of content.

See also:

[mail_load_message\(\)](#)

Parameters:

meta the meta message object of the message to be loaded from disk.

user the meta user object of the user that owns the requested message.

server the server object of the web server where the spam teacher application is hosted.

lines the maximum number of lines to be retrieved from the loaded message.

parse sets the parse parameter passed to [mail_load_message\(\)](#).

Returns:

NULL on failure or a a mail message object containing the retrieved mail message data (truncated if necessary) on success.

Definition at line 278 of file load_message.c.

References length, mail_load_message(), st_char_get(), st_length_get(), st_length_set(), and mail_message_t::text.

Referenced by pop_top().

5.197 src/objects/mail/mail.h File Reference

Data Structures

- struct [mail_cache_t](#)
- struct [mail_mime_t](#)
- struct [mail_message_t](#)
- struct [media_type_t](#)

Defines

- #define [MAIL_MIME_RECURSION_LIMIT](#) 16
- #define [MAIL_SIGNATURES_RECURSION_LIMIT](#) 16

Functions

- void [mail_cache_destroy](#) (void *holder)
cache.c
- [stringer_t](#) * [mail_cache_get](#) (uint64_t messagenum)
Attempt to retrieve the contents of a message from the thread's cached data.
- void [mail_cache_reset](#) (void)
Reset the thread's mail cache and free any held message.
- void [mail_cache_set](#) (uint64_t messagenum, [stringer_t](#) *text)
Set the contents of the thread's mail cache.
- [bool_t](#) [mail_cache_start](#) (void)
Initialize the thread-specific data key used for the mail message cache.
- void [mail_cache_stop](#) (void)
Destroy the thread-specific data key used for the mail message cache.
- void [mail_cache_thread_stop](#) (void)
Free the thread-specific data for the mail message cache.
- void [mail_destroy_header](#) ([stringer_t](#) *header)
cleanup.c
- [bool_t](#) [mail_message_cleanup](#) ([stringer_t](#) **message)
Clean up the body of a message read in via smtp, enforcing magma.smtp.wrap_line_length.
- uint32_t [mail_count_received](#) ([stringer_t](#) *message)
counters.c
- size_t [mail_header_end](#) ([stringer_t](#) *message)
Get the number of bytes taken up by a mail header.
- [bool_t](#) [mail_db_delete_message](#) (uint64_t usernum, uint64_t messagenum, uint32_t size, [int_t](#) transaction)
datatier.c

- void [mail_db_hide_message](#) (uint64_t messagenum)
Set a message invisible in the database.
- uint64_t [mail_db_insert_duplicate_message](#) (uint64_t usernum, uint64_t foldernum, uint32_t status, uint32_t size, uint64_t signum, uint64_t sigkey, uint64_t created, int_t transaction)
Insert a duplicate entry for a message in the database.
- uint64_t [mail_db_insert_message](#) (uint64_t usernum, uint64_t foldernum, uint32_t status, uint32_t size, uint64_t signum, uint64_t sigkey, int_t transaction)
Insert a mail message into the database.
- int_t [mail_db_update_message_folder](#) (uint64_t usernum, uint64_t messagenum, uint64_t source, uint64_t target, int64_t transaction)
Update a mail message's parent folder in the database.
- void [mail_add_forward_headers](#) (server_t *server, stringer_t **message, stringer_t *id, int_t mark, uint64_t signum, uint64_t sigkey)
headers.c
- stringer_t * [mail_add_inbound_headers](#) (connection_t *con, smtp_inbound_prefs_t *prefs)
Prepend the Return-Path: and Received: headers to an inbound smtp message.
- int_t [mail_add_outbound_headers](#) (connection_t *con)
Add a Received: header and dkim signature to an outbound relayed smtp message.
- bool_t [mail_add_required_headers](#) (connection_t *con, smtp_message_t *message)
Generate any missing required headers for an smtp message object.
- stringer_t * [mail_header_fetch_all](#) (stringer_t *header, stringer_t *key)
Get all the values that match a named header line in a message header.
- stringer_t * [mail_header_fetch_cleaned](#) (stringer_t *header, stringer_t *key)
Fetch the value of a specified line from a mail header, performing whitespace and line cleaning.
- placer_t [mail_header_pop](#) (stringer_t *header, size_t *position)
Pop the next header line from a mail message header.
- bool_t [mail_headers](#) (smtp_message_t *message)
Parse an smtp message header and store the To, From, Date, and Subject header values.
- void [mail_mod_subject](#) (stringer_t **message, chr_t *label)
Prepend text to a Subject header line inside of a mail message.
- placer_t [mail_store_header](#) (chr_t *stream, size_t length)
Return a placer pointing to the trimmed value of a mail header.
- stringer_t * [mail_load_header](#) (meta_message_t *meta, meta_user_t *user, server_t *server, bool_t parse)
load_message.c
- mail_message_t * [mail_load_message](#) (meta_message_t *meta, meta_user_t *user, server_t *server, bool_t parse)
Load a stored mail message from disk.
- mail_message_t * [mail_load_message_top](#) (meta_message_t *meta, meta_user_t *user, server_t *server, uint64_t lines, bool_t parse)
Get the top of a mail message, up to a specified maximum number of lines of content.

- [stringer_t * mail_mime_boundary](#) ([placer_t](#) header)
mime.c
- [placer_t mail_mime_child](#) ([placer_t](#) body, [stringer_t](#) *boundary, [uint32_t](#) child)
Get a placer pointing to the specified child inside a MIME body.
- [stringer_t * mail_mime_content_encoding](#) ([placer_t](#) header)
Get the content encoding type from a mime header.
- [stringer_t * mail_mime_content_id](#) ([placer_t](#) header)
Get the content id from a mime header.
- [uint32_t mail_mime_count](#) ([placer_t](#) body, [stringer_t](#) *boundary)
Count the number of instances of a boundary string inside a MIME body.
- [int_t mail_mime_encoding](#) ([placer_t](#) header)
Get the encoding type from a MIME header.
- [void mail_mime_free](#) ([mail_mime_t](#) *mime)
Free a mail mime object and its underlying data, and recursively free its children parts.
- [placer_t mail_mime_header](#) ([stringer_t](#) *part)
Get a placer pointing to a mime header in a mime part.
- [mail_mime_t * mail_mime_part](#) ([stringer_t](#) *part, [uint32_t](#) recursion)
Parse a block of data into a mail mime object.
- [array_t * mail_mime_split](#) ([placer_t](#) body, [stringer_t](#) *boundary)
Split a mime body into an array of children by a boundary string.
- [int_t mail_mime_type](#) ([placer_t](#) header)
Get the content type from a MIME header.
- [stringer_t * mail_mime_type_group](#) ([placer_t](#) header)
Get the value of the Content-Type header from a mime header.
- [array_t * mail_mime_type_parameters](#) ([placer_t](#) header)
Get an array of the key/value pairs of parameters passed to the value of the mime Content-Type header.
- [stringer_t * mail_mime_type_parameters_key](#) ([stringer_t](#) *parameter)
Get the key name of a mime header line parameter.
- [stringer_t * mail_mime_type_parameters_value](#) ([stringer_t](#) *parameter)
Get the value of a mime header line parameter.
- [stringer_t * mail_mime_type_sub](#) ([placer_t](#) header)
Get the subtype of the Content-Type header value from a mime header.
- [int_t mail_mime_update](#) ([mail_message_t](#) *message)
Re-parse a mail message's data as a mime part, freeing any existing mime part(s) that may have already existed.
- [media_type_t * mail_mime_get_media_type](#) ([chr_t](#) *extension)
Get the media type for a given file extension.

- [stringer_t * mail_mime_generate_boundary](#) ([array_t *parts](#))
Generate a MIME boundary string that is unique to a collection of content.
- [stringer_t * mail_mime_encode_part](#) ([stringer_t *data](#), [stringer_t *filename](#), [stringer_t *boundary](#))
Encode a MIME part for a provided block of data (file attachment) with the specified filename.
- [stringer_t * mail_mime_get_smtp_envelope](#) ([stringer_t *from](#), [inx_t *tos](#), [inx_t *ccs](#), [inx_t *bccs](#), [stringer_t *subject](#), [stringer_t *boundary](#), [bool_t attached](#))
Get smtp envelope data for an outbound message sent by a webmail client.
- [mail_message_t * mail_message](#) ([stringer_t *text](#))
objects.c
- [smtp_message_t * mail_create_message](#) ([stringer_t *text](#))
Create an smtp message object out of a buffer of raw data supplied via the smtp DATA command.
- [void mail_destroy](#) ([mail_message_t *message](#))
Free a mail message and all its underlying data.
- [void mail_destroy_message](#) ([smtp_message_t *message](#))
Destroy an smtp message and free all of its underlying data.
- [stringer_t * mail_extract_address](#) ([stringer_t *address](#))
parsing.c
- [placer_t * mail_domain_get](#) ([stringer_t *address](#), [placer_t *output](#))
Get the domain portion of an email address.
- [chr_t * mail_message_path](#) ([uint64_t number](#), [chr_t *server](#))
paths.c
- [bool_t mail_create_directory](#) ([uint64_t number](#), [chr_t *server](#))
Create the on-disk directory structure necessary to hold a given message's file data.
- [int_t mail_path_finder](#) ([chr_t *string](#))
- [bool_t mail_remove_message](#) ([uint64_t usernum](#), [uint64_t messagenum](#), [uint32_t size](#), [chr_t *server](#))
remove_message.c
- [stringer_t * mail_build_signature](#) ([server_t *server](#), [int_t content_type](#), [int_t content_encoding](#), [uint64_t signum](#), [uint64_t sigkey](#), [int_t disposition](#))
signatures.c
- [int_t mail_discover_encoding](#) ([stringer_t *header](#))
Get the value of the Content-Transfer-Encoding header in a mail message header.
- [size_t mail_discover_insertion_point](#) ([stringer_t *message](#), [stringer_t *part](#), [int_t type](#))
Find the position in a mail message where a custom message can be inserted.
- [int_t mail_discover_type](#) ([stringer_t *header](#))
Get the value of the Content-Type header in a mail message header.
- [stringer_t * mail_extract_tag](#) ([chr_t *stream](#), [size_t length](#))

Extract an html tag from a data buffer, searching backwards.

- `stringer_t * mail_get_boundary (stringer_t *header)`

Get the boundary string from a message header.

- `placer_t mail_get_chunk (stringer_t *message, stringer_t *boundary, int_t chunk, bool_t *last)`

Get a specified chunk (mime part) of a multipart mime message.

- `stringer_t * mail_insert_chunk_base64 (server_t *server, stringer_t *message, stringer_t *part, uint64_t signum, uint64_t sigkey, int_t disposition, int_t type, int_t encoding)`

Insert a spam signature training link into a base64-encoded message part.

- `stringer_t * mail_insert_chunk_text (server_t *server, stringer_t *message, stringer_t *part, uint64_t signum, uint64_t sigkey, int_t disposition, int_t type, int_t encoding)`

Insert a spam signature training link into a plain text message part.

- `int_t mail_modify_part (server_t *server, mail_message_t *message, stringer_t *part, uint64_t signum, uint64_t sigkey, int_t disposition, int_t recursion)`

Insert a spam signature training link into a specified part of a mime message.

- `void mail_signature_add (mail_message_t *message, server_t *server, uint64_t signum, uint64_t sigkey, int_t disposition)`

Insert a spam signature training link into a mail message.

- `uint64_t mail_copy_message (uint64_t usernum, uint64_t original, chr_t *server, uint32_t size, uint64_t foldernum, uint32_t status, uint64_t signum, uint64_t sigkey, uint64_t created)`

store_message.c

- `int_t mail_move_message (uint64_t usernum, uint64_t messagenum, uint64_t source, uint64_t target)`

Move a message to a new folder in the database.

- `uint64_t mail_store_message (uint64_t usernum, prime_t *signet, uint64_t foldernum, uint32_t *status, uint64_t signum, uint64_t sigkey, stringer_t *message)`

Store a mail message, with its meta-information in the database, and the contents persisted to disk.

- `bool_t mail_store_message_data (uint64_t messagenum, uint8_t fflags, stringer_t *data, chr_t **pathptr)`

Persist a message's data to disk.

5.197.1 Define Documentation

5.197.1.1 #define MAIL_MIME_RECURSION_LIMIT 16

Definition at line 11 of file mail.h.

Referenced by mail_mime_part(), and portal_message_body().

5.197.1.2 #define MAIL_SIGNATURES_RECURSION_LIMIT 16

Definition at line 12 of file mail.h.

Referenced by mail_modify_part().

5.197.2 Function Documentation

5.197.2.1 void mail_add_forward_headers (server_t * server, stringer_t ** message, stringer_t * id, int_t mark, uint64_t signum, uint64_t sigkey)

headers.c headers.c

Note:

This function will prepend tags to the Subject line like "JUNK", "INFECTED", "SPOOFED", etc. The following headers will be stripped from the message: Sender, Return-Path, DKIM-Signature, and DomainKey-Signature. This daemon will be reinserted into the headers as the origin of the Sender: header value. If signum and sigkey are both set, then a spam training url will also be inserted into the message. Finally, if [magma.dkim.enabled](#) is set, a dkim signature will be added to the message.

Parameters:

server the server object of the web server where the teacher application is hosted.

message the address of a managed string containing the message data that will be set to the modified messagee data on success.

id a managed string containing the message id (for dkim).

mark a bitmask of marks to be tagged in the message subject (SMTP_MARK_SPAM, SMTP_MARK_VIRUS, SMTP_MARK_SPOOF, SMTP_MARK_RBL, and SMTP_MARK_PHISH).

signum the optional spam signature number referenced by the teacher url.

sigkey the optional spam signature key for client verification in the teacher app.

Returns:

This function returns no value. LOW: We should be passing the message as a stringer_t * and then returning it as a stringer_t *.

Definition at line 730 of file headers.c.

References CONSTANT, magma_t::dkim, dkim_signature_create(), magma_t::enabled, HEAP, JOINED, log_pedantic, magma, mail_destroy(), mail_header_end(), mail_header_pop(), MAIL_MARK_JUNK, mail_message(), mail_mod_subject(), mail_signature_add(), MAPPED_T, mm_cmp_ci_eq(), pl_empty(), PLACER, placer_t, SMTP_MARK_PHISH, SMTP_MARK_RBL, SMTP_MARK_SPAM, SMTP_MARK_SPOOF, SMTP_MARK_VIRUS, st_alloc_opts(), st_append_opts(), st_char_get(), st_cleanup, st_cmp_ci_starts(), st_cmp_cs_starts(), st_data_get(), st_dupe(), st_dupe_opts(), st_free(), st_length_get(), st_merge_opts(), status, and mail_message_t::text.

Referenced by smtp_forward_message().

5.197.2.2 stringer_t* mail_add_inbound_headers (connection_t * con, smtp_inbound_prefs_t * prefs)

Prepend the Return-Path: and Received: headers to an inbound smtp message.

Parameters:

con the connection across which the inbound smtp message was accepted.

prefs the smtp inbound preferences corresponding to the connection, with the rcptto field populated.

Returns:

NULL on failure, or a managed string containing the message data preceded by the Return-Path and Received headers on success.

Definition at line 638 of file headers.c.

References con_addr_presentation(), con_reverse_check(), CONSTANT, HEAP, JOINED, log_pedantic, MANAGEDBUF, MAPPED_T, smtp_inbound_prefs_t::rcptto, st_cmp_ci_eq(), st_cmp_cs_eq(), and st_merge_opts().

Referenced by smtp_accept_message().

5.197.2.3 int_t mail_add_outbound_headers (connection_t * con)

Add a Received: header and dkim signature to an outbound relayed smtp message.

Note:

If the message already has a Received header, the dkim signature will be inserted right before the first instance.

Parameters:

con the connection across which the outbound smtp message is being sent.

Returns:

NULL on failure, or a managed string containing the message data preceded by the Received header and including the dkim signature on success.

Definition at line 849 of file headers.c.

References con_addr_presentation(), con_reverse_check(), magma_t::dkim, dkim_signature_create(), domain_dkim(), magma_t::enabled, HEAP, JOINTED, log_pedantic, magma, mail_domain_get(), mail_header_end(), mail_header_pop(), MANAGEDBUF, MAPPED_T, mm_cmp_ci_eq(), pl_empty(), PLACER, placer_t, st_char_get(), st_cleanup, st_cmp_cs_eq(), st_data_get(), st_dupe(), st_empty, st_free(), st_import(), st_length_get(), and st_merge_opts().

Referenced by smtp_relay_message().

5.197.2.4 bool_t mail_add_required_headers (connection_t * con, smtp_message_t * message)

Generate any missing required headers for an smtp message object.

Note:

These required headers include To, From, Subject and Date, and if they are absent, they will be generated from the specified connection's inbound or outbound preferences, accordingly.

Parameters:

con a pointer to the connection object across which the smtp message was received.

message the smtp message object to be updated for any missing required headers.

Returns:

true on success or false on failure.

Definition at line 457 of file headers.c.

References smtp_recipients_t::address, smtp_message_t::date, smtp_message_t::from, smtp_message_t::header_length, HEAP, JOINTED, length, log_pedantic, mail_header_end(), mail_headers(), MAPPED_T, smtp_recipients_t::next, smtp_inbound_prefs_t::next, PLACER, smtp_inbound_prefs_t::rcptto, st_alloc(), st_avail_get(), st_char_get(), st_cleanup, st_empty, st_free(), st_import(), st_length_get(), st_length_set(), st_merge, st_merge_opts(), smtp_message_t::subject, smtp_message_t::text, and smtp_message_t::to.

Referenced by smtp_data().

5.197.2.5 stringer_t* mail_build_signature (server_t * server, int_t content_type, int_t content_encoding, uint64_t signum, uint64_t sigkey, int_t disposition)

[signatures.c](#) [signatures.c](#)

Parameters:

server the server object of the web server where the teacher application is hosted.

type MESSAGE_TYPE_HTML to generate an html-based signature, or any other value for plain text.

content_encoding if MESSAGE_ENCODING_QUOTED_PRINTABLE, set the encoding type to qp.

signum the spam signature number referenced by the teacher url.

sigkey the spam signature key for client verification in the teacher app.

disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".

Returns:

NULL on failure, or a newly allocated managed string containing the desired mail signature on success.

Definition at line 20 of file signatures.c.

References server_t::domain, length, log_pedantic, MESSAGE_ENCODING_QUOTED_PRINTABLE, MESSAGE_TYPE_HTML, qp_encode(), st_alloc(), st_char_get(), st_cleanup, st_free(), st_length_get(), st_merge, st_sprint(), and uint64_digits().

Referenced by mail_insert_chunk_base64(), and mail_insert_chunk_text().

5.197.2.6 void mail_cache_destroy (void * *holder*)

cache.c cache.c

Parameters:

holder a pointer to the cached mail message object to be freed.

Returns:

This function returns no value.

Definition at line 17 of file cache.c.

References mm_free(), st_cleanup, and mail_cache_t::text.

Referenced by mail_cache_start(), and mail_cache_thread_stop().

5.197.2.7 stringer_t* mail_cache_get (uint64_t *messagenum*)

Attempt to retrieve the contents of a message from the thread's cached data.

Note:

The thread's cache only has the capacity to store a single message.

Parameters:

messagenum the id of the message to be retrieved.

Returns:

NULL on failure or a managed string containing the message data on success.

Definition at line 78 of file cache.c.

References mail_cache_t::messagenum, st_dupe(), and mail_cache_t::text.

Referenced by mail_load_message().

5.197.2.8 void mail_cache_reset (void)

Reset the thread's mail cache and free any held message.

Returns:

This function returns no value.

Definition at line 97 of file cache.c.

References mail_cache_t::messagenum, mm_free(), st_cleanup, and mail_cache_t::text.

Referenced by imap_session_destroy(), and pop_session_destroy().

5.197.2.9 void mail_cache_set (uint64_t messagenum, stringer_t * text)

Set the contents of the thread's mail cache.

Parameters:

messagenum the numerical id of the message to be cached. a managed string containing the contents of the specified message to be cached.

Returns:

This function returns no value.

Definition at line 118 of file cache.c.

References CONTIGUOUS, HEAP, log_pedantic, MANAGED_T, mail_cache_t::messagenum, mm_alloc(), mm_free(), st_cleanup, st_dup_opts(), and mail_cache_t::text.

Referenced by mail_load_message().

5.197.2.10 bool_t mail_cache_start (void)

Initialize the thread-specific data key used for the mail message cache.

Returns:

true on success or false on failure.

Definition at line 33 of file cache.c.

References log_pedantic, and mail_cache_destroy().

Referenced by process_start().

5.197.2.11 void mail_cache_stop (void)

Destroy the thread-specific data key used for the mail message cache.

Returns:

This function returns no value.

Definition at line 47 of file cache.c.

References log_pedantic.

Referenced by process_stop().

5.197.2.12 void mail_cache_thread_stop (void)

Free the thread-specific data for the mail message cache.

Returns:

This function returns no value.

Definition at line 60 of file cache.c.

References mail_cache_destroy().

Referenced by thread_stop().

5.197.2.13 uint64_t mail_copy_message (uint64_t usernum, uint64_t original, chr_t * server, uint32_t size, uint64_t foldernum, uint32_t status, uint64_t signum, uint64_t sigkey, uint64_t created)

[store_message.c](#) [store_message.c](#)

Parameters:

usernum the numerical id of the user to whom the mail message belongs.

original the numerical id of the mail message to be copied.

server a pointer to a null-terminated string containing the name of the server where the message contents are stored.

size the size, in bytes, of the mail message to be copied.

foldernum the numerical id of the folder to become the parent folder of the message copy.

signum the spam signature for the message.

sigkey the spam key for the message.

created the UNIX timestamp of when the message was created.

Returns:

0 on failure, or the ID of the copy of the mail message in the database on success.

Definition at line 195 of file store_message.c.

References log_error, log_pedantic, mail_create_directory(), mail_db_insert_duplicate_message(), mail_message_path(), ns_free(), tran_commit(), tran_rollback(), and tran_start().

Referenced by imap_message_copier(), and meta_messages_copier().

5.197.2.14 uint32_t mail_count_received (stringer_t * message)

[counters.c](#) [counters.c](#)

Parameters:

message a managed string containing the mail message to be scanned.

Returns:

the number of matching lines found, or 0 on failure.

Definition at line 15 of file counters.c.

References log_pedantic, PLACER, st_cmp_ci_starts(), and st_empty_out().

Referenced by smtp_data().

5.197.2.15 bool_t mail_create_directory (uint64_t *number*, chr_t * *server*)

Create the on-disk directory structure necessary to hold a given message's file data.

Parameters:

number the mail message id.

server the hostname of the server where the message data resides or if NULL, the default server.

Returns:

true on success or false on failure.

Definition at line 66 of file paths.c.

References magma_t::active, log_error, magma, magma_t::root, st_char_get(), st_length_int(), and magma_t::storage.

Referenced by mail_copy_message(), and mail_store_message_data().

5.197.2.16 smtp_message_t* mail_create_message (stringer_t * *text*)

Create an smtp message object out of a buffer of raw data supplied via the smtp DATA command.

Note:

This function will also generate a random message ID.

Parameters:

text a pointer to a managed string containing the smtp data to be parsed.

Returns:

NULL on failure or a pointer to the newly initialized smtp message object wrapping the data on success.

Definition at line 127 of file objects.c.

References smtp_message_t::header_length, smtp_message_t::id, log_pedantic, mail_header_end(), mail_headers(), mm_alloc(), mm_free(), rand_choices(), st_free(), and smtp_message_t::text.

Referenced by smtp_data().

5.197.2.17 bool_t mail_db_delete_message (uint64_t *usernum*, uint64_t *messagenum*, uint32_t *size*, int_t *transaction*)

datatier.c datatier.c

Parameters:

usernum the user id to whom the target mail message belongs.

messagenum the message id of the mail message to be deleted.

size the storage size, in bytes, of the message to be deleted.

transaction the mysql connection id on which to execute the statements.

Returns:

0 on failure or 1 on success.

Definition at line 41 of file datatier.c.

References log_error, mm_wipe(), and stmt_exec_affected_conn().

Referenced by mail_remove_message().

5.197.2.18 void mail_db_hide_message (uint64_t *messagenum*)

Set a message invisible in the database.

Parameters:

messagenum the message id of the mail message to be hidden.

Returns:

This function returns no value.

Definition at line 15 of file `datatier.c`.

References `mm_wipe()`, and `stmt_exec()`.

Referenced by `mail_load_message()`.

5.197.2.19 uint64_t mail_db_insert_duplicate_message (uint64_t *usernum*, uint64_t *foldernum*, uint32_t *status*, uint32_t *size*, uint64_t *signum*, uint64_t *sigkey*, uint64_t *created*, int_t *transaction*)

Insert a duplicate entry for a message in the database.

Note:

This function will also update the user's storage quota information in the database.

Parameters:

usernum the numerical id of the user that owns the message.

foldernum the numerical id of the parent folder containing the message.

status the status flags value for the message.

size the size, in bytes, of the mail message on disk.

signum the spam signature for the message.

sigkey the spam key for the message.

created the UNIX timestamp of when the message was created.

transaction the transaction id for the database operation, in case the caller wants to roll back the transaction.

Returns:

NULL on failure, or the ID of the newly inserted message on success.

Definition at line 304 of file `datatier.c`.

References `magma_t::active`, `ISNULL`, `log_pedantic`, `magma`, `MAIL_MARK_JUNK`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `stmt_exec_conn()`, `stmt_insert_conn()`, and `magma_t::storage`.

Referenced by `mail_copy_message()`.

5.197.2.20 uint64_t mail_db_insert_message (uint64_t *usernum*, uint64_t *foldernum*, uint32_t *status*, uint32_t *size*, uint64_t *signum*, uint64_t *sigkey*, int_t *transaction*)

Insert a mail message into the database.

Note:

This function will also update the user's storage quota information in the database.

Parameters:

usenum the numerical id of the user to whom the mail message will belong.

foldernum the numerical id of the folder that will be the parent folder of the message.

status the status flags of the mail message.

size the size, in bytes, of the mail message on disk.

signum the spam signature for the message.

sigkey the spam key for the message.

transaction the transaction id for the database operation, in case the caller wants to roll back the transaction.

Returns:

0 on failure or the id of the newly inserted mail message on success.

Definition at line 172 of file `datatier.c`.

References `magma_t::active`, `ISNULL`, `log_pedantic`, `magma`, `MAIL_MARK_JUNK`, `mm_wipe()`, `st_char_get()`, `st_length_get()`, `stmt_exec_conn()`, `stmt_insert_conn()`, and `magma_t::storage`.

Referenced by `mail_store_message()`.

5.197.2.21 `int_t mail_db_update_message_folder (uint64_t usenum, uint64_t messagenum, uint64_t source, uint64_t target, int64_t transaction)`

Update a mail message's parent folder in the database. *usenum* the numerical id of the user to whom the mail message belongs. *messagenum* the numerical id of the target mail message of the operation. *source* the numerical id of the parent folder in which the mail message currently resides. *target* the numerical id of the destination folder which is to be the new parent of the mail message. *transaction* a transaction id for the database operation, in case the caller needs to roll back changes on failure.

Returns:

-1 on failure, 0 if the target message could not be located in the database, or 1 on success.

Definition at line 100 of file `datatier.c`.

References `log_pedantic`, `mm_wipe()`, `mysql_stmt_errno_d`, `mysql_stmt_error_d`, `pool_get_obj()`, `sql_pool`, and `stmt_exec_affected_conn()`.

Referenced by `mail_move_message()`.

5.197.2.22 `void mail_destroy (mail_message_t * message)`

Free a mail message and all its underlying data.

Parameters:

message a pointer to the mail message object to be freed.

Returns:

This function returns no value.

Definition at line 32 of file `objects.c`.

References `mail_message_t::from`, `mail_mime_free()`, `mail_message_t::mime`, `mm_free()`, `st_cleanup`, and `mail_message_t::text`.

Referenced by `imap_fetch_body()`, `imap_fetch_message()`, `imap_fetch_return_header()`, `imap_fetch_return_message()`, `imap_fetch_return_mime()`, `imap_fetch_return_text()`, `imap_search_messages()`, `mail_add_forward_headers()`, `mail_load_header()`, `pop_retr()`, `pop_top()`, and `portal_endpoint_messages_load()`.

5.197.2.23 void mail_destroy_header (stringer_t * header)[cleanup.c cleanup.c](#)**Parameters:**

header a managed string containing the mail header to be freed.

Returns:

This function returns no value.

Definition at line 15 of file cleanup.c.

References st_cleanup.

Referenced by imap_fetch_body(), imap_fetch_message(), imap_fetch_return_message(), imap_fetch_return_mime(), imap_fetch_return_text(), and imap_search_messages().

5.197.2.24 void mail_destroy_message (smtp_message_t * message)

Destroy an smtp message and free all of its underlying data.

Parameters:

message a pointer to the smtp message to be destroyed.

Returns:

This function returns no value.

Definition at line 15 of file objects.c.

References smtp_message_t::from, smtp_message_t::id, mm_free(), st_cleanup, and smtp_message_t::text.

Referenced by smtp_data(), smtp_session_destroy(), and smtp_session_reset().

5.197.2.25 int_t mail_discover_encoding (stringer_t * header)

Get the value of the Content-Transfer-Encoding header in a mail message header.

Note:

Possible return values include MESSAGE_ENCODING_QUOTED_PRINTABL, MESSAGE_ENCODING_BASE64, MESSAGE_ENCODING_8BIT, and MESSAGE_ENCODING_7BIT.

Parameters:

header a managed string containing the mail message header to be parsed.

Returns:

the MESSAGE_ENCODING code of the mail transfer encoding type, or MESSAGE_ENCODING_7BIT by default.

Definition at line 171 of file signatures.c.

References content, mail_header_fetch_all(), MESSAGE_ENCODING_7BIT, MESSAGE_ENCODING_8BIT, MESSAGE_ENCODING_BASE64, MESSAGE_ENCODING_QUOTED_PRINTABLE, MESSAGE_ENCODING_UNKNOWN, mm_cmp_ci_eq(), PLACER, st_char_get(), st_free(), and st_length_get().

Referenced by mail_modify_part().

5.197.2.26 `size_t mail_discover_insertion_point (stringer_t * message, stringer_t * part, int_t type)`

Find the position in a mail message where a custom message can be inserted.

Note:

The part parameter is expected to be a placer pointing into the contents of message. If the encoding type is not html, the insertion point is determined to be at the end of the part. If the encoding type is html, the following rules are followed: 1. Scanning backwards from the end of the message, skip trailing whitespace get the next html tag. 2. If that tag is NOT `</html>` or `</body>`, insert the signature AFTER it. 3. If that tag IS `</html>` or `</body>`, insert the signature right before it closes.

Parameters:

message a managed string containing the mail message body to be parsed.

part a managed string (placer) containing the part of the message where the custom message should be inserted.

type the encoding type of the message (MESSAGE_TYPE_HTML or other).

Returns:

the zero-based index of the position in the specified message where the custom message can be inserted.

Definition at line 294 of file signatures.c.

References `length`, `mail_extract_tag()`, `MESSAGE_TYPE_HTML`, `PLACER`, `st_char_get()`, `st_cmp_ci_eq()`, `st_data_get()`, `st_free()`, and `st_length_get()`.

Referenced by `mail_insert_chunk_base64()`, and `mail_insert_chunk_text()`.

5.197.2.27 `int_t mail_discover_type (stringer_t * header)`

Get the value of the Content-Type header in a mail message header.

Note:

Possible return values include `MESSAGE_TYPE_MULTI_ALTERNATIVE`, `MESSAGE_TYPE_MULTI_RELATED`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_HTML`, `MESSAGE_TYPE_MULTI_UNKOWN`, and `MESSAGE_TYPE_PLAIN`.

Parameters:

header a managed string containing the mail message header to be parsed.

Returns:

the MESSAGE_TYPE code of the mail content type, or MESSAGE_TYPE_PLAIN by default.

Definition at line 120 of file signatures.c.

References `content`, `mail_header_fetch_all()`, `MESSAGE_TYPE_HTML`, `MESSAGE_TYPE_MULTI_ALTERNATIVE`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_MULTI_RELATED`, `MESSAGE_TYPE_MULTI_UNKOWN`, `MESSAGE_TYPE_PLAIN`, `mm_cmp_ci_eq()`, `PLACER`, `st_char_get()`, `st_free()`, and `st_length_get()`.

Referenced by `mail_modify_part()`.

5.197.2.28 `placer_t* mail_domain_get (stringer_t * address, placer_t * output)`

Get the domain portion of an email address.

Parameters:

address a managed string containing the email address to be parsed.

output a pointer to a placer to receive the domain portion of the email address.

Returns:

NULL on failure or a pointer to the output of the domain extraction on success.

Definition at line 16 of file parsing.c.

References tok_get_count_st(), and tok_get_st().

Referenced by mail_add_outbound_headers(), smtp_check_authorized_from(), smtp_fetch_inbound(), smtp_rcpt_to(), and spf_check().

5.197.2.29 stringer_t* mail_extract_address (stringer_t * address)

[parsing.c](#) [parsing.c](#)

Note:

The preference of this function is first to try to find the email addressed enclosed in <>. No valid email address may be enclosed in () or "".

Parameters:

address a managed string containing the buffer from which the email address will be extracted.

Returns:

NULL on failure or a managed string containing the email address on success.

Definition at line 32 of file parsing.c.

References comment, length, log_pedantic, st_alloc(), st_char_get(), st_data_get(), st_empty_out(), and st_length_set().

Referenced by smtp_data_outbound().

5.197.2.30 stringer_t* mail_extract_tag (chr_t * stream, size_t length)

Extract an html tag from a data buffer, searching backwards.

Warning:

Remember that this function takes the end and not the start of a buffer!

Note:

The extracted tag contains the enclosing brackets and only printable characters (no whitespace).

Parameters:

stream the end of a data buffer containing the html data to be parsed.

length the size, in bytes, of the data buffer to be parsed.

Returns:

NULL on failure, or a managed string containing the printable contents of the nearest html tag on success.

Definition at line 218 of file signatures.c.

References log_pedantic, st_alloc(), st_char_get(), st_free(), and st_length_set().

Referenced by mail_discover_insertion_point().

5.197.2.31 stringer_t* mail_get_boundary (stringer_t * *header*)

Get the boundary string from a message header.

Note:

This function works by parsing the Content-Type header if it exists, falling back to the entire header otherwise. The returned boundary string includes a trailing "--".

Parameters:

header a managed string containing the message header.

Returns:

NULL on failure or a managed string containing the boundary string on success.

Definition at line 643 of file signatures.c.

References content, length, log_pedantic, mail_header_fetch_all(), PLACER, st_char_get(), st_cleanup, st_data_get(), st_length_get(), st_merge, and st_search_ci().

Referenced by mail_modify_part().

5.197.2.32 placer_t mail_get_chunk (stringer_t * *message*, stringer_t * *boundary*, int_t *chunk*, bool_t * *last*)

Get a specified chunk (mime part) of a multipart mime message.

Parameters:

result a pointer to the placer that will contain the result

message a managed string containing the mime message to be parsed.

boundary a managed string containing the boundary used to split the multipart mime message.

chunk the one-index based chunk to be retrieved from the multipart message

Returns:

NULL on failure or a pointer to the placer containing the specified chunk.

Definition at line 543 of file signatures.c.

References length, log_error, log_pedantic, mm_cmp_cs_eq(), pl_init(), pl_null(), PLACER, st_char_get(), st_cmp_cs_eq(), st_length_get(), and st_search_cs().

Referenced by mail_modify_part().

5.197.2.33 size_t mail_header_end (stringer_t * *message*)

Get the number of bytes taken up by a mail header.

Parameters:

message a managed string containing the full smtp mail message.

Returns:

0 on failure, or the number of bytes occupied by the mail header.

Definition at line 56 of file counters.c.

References length, log_pedantic, st_char_get(), and st_length_get().

Referenced by mail_add_forward_headers(), mail_add_outbound_headers(), mail_add_required_headers(), mail_create_message(), mail_headers(), mail_message(), mail_mod_subject(), naked_message_set(), and smtp_check_filters().

5.197.2.34 stringer_t* mail_header_fetch_all (stringer_t * *header*, stringer_t * *key*)

Get all the values that match a named header line in a message header.

Parameters:

header a managed string containing the message header.

key a managed string containing the header name to be searched for.

Returns:

NULL on failure or a managed string containing all the matching header line values separated by newlines.

Definition at line 147 of file headers.c.

References mm_cmp_ci_eq(), PLACER, st_char_get(), st_empty, st_free(), st_import(), st_length_get(), and st_merge.

Referenced by imap_fetch_body_mime(), imap_search_messages_header(), mail_discover_encoding(), mail_discover_type(), mail_get_boundary(), mail_mime_boundary(), and smtp_check_filters().

5.197.2.35 stringer_t* mail_header_fetch_cleaned (stringer_t * *header*, stringer_t * *key*)

Fetch the value of a specified line from a mail header, performing whitespace and line cleaning.

Note:

This function handles mail header values that span new lines, although the output excludes line breaks and consecutive space characters.

Parameters:

header a managed string containing the full header of the target mail message.

key a managed string containing the name of the desired header name to be extracted, excluding the trailing ":"

Returns:

NULL on failure, or a managed string containing the cleaned value of the specified mail header line on success.

Definition at line 17 of file headers.c.

References PLACER, st_alloc(), st_char_get(), st_cmp_ci_starts(), st_empty, st_length_get(), st_length_set(), and st_trim().

Referenced by imap_fetch_bodystructure(), imap_fetch_envelope(), imap_search_messages_date(), mail_headers(), mail_message(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_encoding(), mail_mime_type(), mail_mime_type_group(), mail_mime_type_parameters(), mail_mime_type_sub(), naked_message_set(), portal_endpoint_messages_list(), and portal_message_header().

5.197.2.36 placer_t mail_header_pop (stringer_t * *header*, size_t * *position*)

Pop the next header line from a mail message header.

Parameters:

header a managed string containing the entire mail message header.

position a pointer to a size_t that contains the tracker index into the entire header for the pop operation, and will also receive the new value of the tracker for the subsequent call, when a new header line is located.

Returns:

[pl_null\(\)](#) on error or a placer pointing to the next mail message subject line on success.

Definition at line 236 of file headers.c.

References pl_init(), pl_null(), st_char_get(), st_empty, and st_length_get().

Referenced by imap_fetch_body_header(), mail_add_forward_headers(), mail_add_outbound_headers(), and mail_mod_subject().

5.197.2.37 bool_t mail_headers (smtp_message_t * message)

Parse an smtp message header and store the To, From, Date, and Subject header values.

Note:

The found header fields will be stored inside the same smtp message object passed by the caller as input.

Parameters:

message a pointer to a partially populated smtp message object that contains the raw message data to be parsed.

Returns:

true on success or false on failure.

Definition at line 318 of file headers.c.

References smtp_message_t::date, smtp_message_t::from, smtp_message_t::header_length, length, log_pedantic, mail_header_end(), mail_header_fetch_cleaned(), mail_store_header(), mm_cmp_ci_eq(), PLACER, st_char_get(), st_free(), st_length_get(), smtp_message_t::subject, smtp_message_t::text, and smtp_message_t::to.

Referenced by mail_add_required_headers(), and mail_create_message().

5.197.2.38 stringer_t* mail_insert_chunk_base64 (server_t * server, stringer_t * message, stringer_t * part, uint64_t signum, uint64_t sigkey, int_t disposition, int_t type, int_t encoding)

Insert a spam signature training link into a base64-encoded message part.

See also:

[mail_discover_insertion_point\(\)](#)

Note:

This is similar to [mail_insert_chunk_text\(\)](#) except the part has to be decoded and then re-encoded after the training link is inserted.

Parameters:

server the server object of the web server where the teacher application is hosted.

message a managed string containing the base64-encoded message body to be parsed.

part a managed string (placer) containing the part of the message where the signature should be inserted.

signum the spam signature number referenced by the teacher url.

sigkey the spam signature key for client verification in the teacher app.

disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".

type the encoding type of the message (MESSAGE_TYPE_HTML or other).

encoding if MESSAGE_ENCODING_QUOTED_PRINTABLE, set the encoding type to qp.

Returns:

NULL on failure or a managed string containing the base64-encoded message with the inserted signature training link on success.

Definition at line 359 of file signatures.c.

References base64_decode(), base64_encode(), length, log_pedantic, mail_build_signature(), mail_discover_insertion_point(), PLACER, st_char_get(), st_cleanup, st_data_get(), st_free(), st_length_get(), and st_merge.

Referenced by mail_modify_part().

5.197.2.39 `stringer_t* mail_insert_chunk_text (server_t * server, stringer_t * message, stringer_t * part, uint64_t signum, uint64_t sigkey, int_t disposition, int_t type, int_t encoding)`

Insert a spam signature training link into a plain text message part.

See also:

[mail_discover_insertion_point\(\)](#)

Parameters:

server the server object of the web server where the teacher application is hosted.

message a managed string containing the message body to be parsed.

part a managed string (placer) containing the part of the message where the signature should be inserted.

signum the spam signature number referenced by the teacher url.

sigkey the spam signature key for client verification in the teacher app.

disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".

type the encoding type of the message (MESSAGE_TYPE_HTML or other).

encoding if MESSAGE_ENCODING_QUOTED_PRINTABLE, set the encoding type to qp.

Returns:

NULL on failure or a managed string containing the message with the inserted signature training link on success.

Definition at line 500 of file signatures.c.

References `log_pedantic`, `mail_build_signature()`, `mail_discover_insertion_point()`, `PLACER`, `st_char_get()`, `st_empty`, `st_free()`, `st_length_get()`, and `st_merge`.

Referenced by `mail_modify_part()`.

5.197.2.40 `stringer_t* mail_load_header (meta_message_t * meta, meta_user_t * user, server_t * server, bool_t parse)`

[load_message.c](#) `load_message.c`

Note:

The file data is first unencrypted and decompressed, according to the file header flags. When extracted, the header's Subject line is branded with any applicable labels such as JUNK, INFECTED, SPOOFED, BLACKHOLED, PHISHING.

Parameters:

meta the meta message object of the message to be queried.

user the meta user object of the user that owns the message.

Returns:

NULL on failure or a managed string containing the message's header on success.

Definition at line 237 of file `load_message.c`.

References `mail_message_t::header_length`, `log_pedantic`, `mail_destroy()`, `mail_load_message()`, `st_char_get()`, `st_import()`, and `mail_message_t::text`.

Referenced by `imap_fetch_return_header()`, `imap_search_messages_date()`, `imap_search_messages_header()`, and `portal_endpoint_messages_list()`.

5.197.2.41 mail_message_t* mail_load_message (meta_message_t * meta, meta_user_t * user, server_t * server, bool_t parse)

Load a stored mail message from disk.

Note:

The mail message will always, at the very least, be compressed using the lzo algorithm; however, on-disk encryption may be enabled. If parsing is enabled, a spam signature training link may be embedded in the message.

Parameters:

meta the meta message object of the message to be loaded from disk.

user the meta user object of the user that owns the requested message.

server the server object of the web server where the spam teacher application is hosted.

parse if true, the header's Subject line is branded with any applicable labels such as JUNK, INFECTED, SPOOFED, BLACKHOLED, PHISHING.

Returns:

NULL on failure or a a mail message object containing the retrieved mail message data on success.

Definition at line 20 of file load_message.c.

References compress_import(), decompress_lzo(), FMESSAGE_MAGIC_1, FMESSAGE_MAGIC_2, FMESSAGE_OPT_COMPRESSED, FMESSAGE_OPT_ENCRYPTED, log_info, log_pedantic, mail_cache_get(), mail_cache_set(), mail_db_hide_message(), MAIL_MARK_BLACKHOLED, MAIL_MARK_INFECTED, MAIL_MARK_JUNK, MAIL_MARK_PHISHING, MAIL_MARK_SPOOFED, mail_message(), mail_message_path(), mail_mod_subject(), mail_signature_add(), MAIL_STATUS_ENCRYPTED, message_header_t, META_USER_ENCRYPT_DATA, ns_free(), OBJECT_MESSAGES, org_signet, prime_message_decrypt(), serial_increment(), st_alloc(), st_char_get(), st_free(), st_length_int(), st_length_set(), and mail_message_t::text.

Referenced by imap_fetch_message(), imap_fetch_return_message(), imap_fetch_return_mime(), imap_fetch_return_text(), imap_search_messages_body(), imap_search_messages_text(), mail_load_header(), mail_load_message_top(), pop_retr(), and portal_endpoint_messages_load().

5.197.2.42 mail_message_t* mail_load_message_top (meta_message_t * meta, meta_user_t * user, server_t * server, uint64_t lines, bool_t parse)

Get the top of a mail message, up to a specified maximum number of lines of content.

See also:

[mail_load_message\(\)](#)

Parameters:

meta the meta message object of the message to be loaded from disk.

user the meta user object of the user that owns the requested message.

server the server object of the web server where the spam teacher application is hosted.

lines the maximum number of lines to be retrieved from the loaded message.

parse sets the parse parameter passed to [mail_load_message\(\)](#).

Returns:

NULL on failure or a a mail message object containing the retrieved mail message data (truncated if necessary) on success.

Definition at line 278 of file load_message.c.

References length, mail_load_message(), st_char_get(), st_length_get(), st_length_set(), and mail_message_t::text.

Referenced by pop_top().

5.197.2.43 mail_message_t* mail_message (stringer_t * text)

objects.c objects.c

Parameters:

text a pointer to the managed string containing the raw (uncompressed) mail data to be parsed.

Returns:

NULL on failure or a pointer to a newly allocated mail message object with the message on success.

Definition at line 54 of file objects.c.

References mail_message_t::date, mail_message_t::from, mail_message_t::header_length, length, log_pedantic, mail_header_end(), mail_header_fetch_cleaned(), mail_store_header(), mm_alloc(), mm_cmp_ci_eq(), mm_free(), PLACER, st_char_get(), st_length_get(), mail_message_t::subject, mail_message_t::text, and mail_message_t::to.

Referenced by mail_add_forward_headers(), and mail_load_message().

5.197.2.44 bool_t mail_message_cleanup (stringer_t ** message)

Clean up the body of a message read in via smtp, enforcing magma.smtp.wrap_line_length.

Note:

This function fixes broken line separators by making sure each is followed by and vice versa. All Return-Path: header lines are also removed. New lines are begun whenever the current length of any line reaches the configuration value set in magma.smtp.wrap_line_length. The trailing dot at the end of the smtp DATA command is also stripped. If the original message ends with , it will have appended to it.

Parameters:

message a pointer to a managed string that contains the message input, and will also store the cleaned output on success.

Returns:

true on success or false on failure.

LOW: Splitting lines, particularly headers causes more problems then it fixes, so this code is currently disabled. At some point we should survey other mail systems and see how they handle long lines.

Definition at line 31 of file cleanup.c.

References HEAP, JOINTED, length, log_pedantic, magma, MAPPED_T, mm_cmp_ci_eq(), magma_t::smtp, st_alloc_opts(), st_char_get(), st_free(), st_length_get(), st_length_set(), and magma_t::wrap_line_length.

Referenced by smtp_data().

5.197.2.45 chr_t* mail_message_path (uint64_t number, chr_t * server)

[paths.c](#) [paths.c](#)

Parameters:

number the mail message id.

server the hostname of the server where the message data resides or if NULL, the default server.

Returns:

NULL on failure, or a pointer to a null-terminated string containing the absolute file path of the specified message.

Definition at line 35 of file paths.c.

References magma_t::active, log_pedantic, magma, ns_alloc(), ns_free(), magma_t::root, st_char_get(), st_length_int(), and magma_t::storage.

Referenced by mail_copy_message(), mail_load_message(), mail_remove_message(), and mail_store_message_data().

5.197.2.46 stringer_t* mail_mime_boundary (placer_t header)

[mime.c](#) [mime.c](#)

Note:

This function first scans the value of Content-Type for the boundary, and then the rest of the MIME header.

Parameters:

header a placer pointing to the MIME header to be parsed.

Returns:

NULL on failure, or a pointer to a managed string containing the MIME boundary string on success.

Definition at line 621 of file mime.c.

References content, length, log_error, log_pedantic, mail_header_fetch_all(), PLACER, st_char_get(), st_cleanup, st_length_get(), st_merge, and st_search_ci().

Referenced by mail_mime_part().

5.197.2.47 placer_t mail_mime_child (placer_t body, stringer_t * boundary, uint32_t child)

Get a placer pointing to the specified child inside a MIME body.

Parameters:

body a placer containing the body text to be parsed.

boundary a pointer to a managed string containing the boundary string to split the MIME content.

child the zero-based index of the MIME child to be located in the body text.

Returns:

[pl_null\(\)](#) on failure, or a placer containing the specified MIME child on success.

Definition at line 753 of file mime.c.

References length, log_pedantic, mm_cmp_cs_eq(), pl_empty(), pl_init(), pl_null(), st_char_get(), st_empty, and st_length_get().

Referenced by mail_mime_split().

5.197.2.48 stringer_t* mail_mime_content_encoding (placer_t header)

Get the content encoding type from a mime header.

Note:

This function parses the Content-Transfer-Encoding field of the header, returning "7bit" by default.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

a managed string containing the content encoding type value of the header, with "7bit" as default.

Definition at line 235 of file mime.c.

References mail_header_fetch_cleaned(), PLACER, st_char_get(), st_free(), st_import(), and st_length_get().

Referenced by imap_fetch_bodystructure().

5.197.2.49 stringer_t* mail_mime_content_id (placer_t header)

Get the content id from a mime header.

Note:

This function parses the Content-Id field of the header, returning "7bit" by default.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

NULL on failure, or a managed string containing the content id value of the header.

Definition at line 281 of file mime.c.

References mail_header_fetch_cleaned(), PLACER, st_char_get(), st_free(), st_import(), and st_length_get().

Referenced by imap_fetch_bodystructure().

5.197.2.50 uint32_t mail_mime_count (placer_t body, stringer_t * boundary)

Count the number of instances of a boundary string inside a MIME body.

Note:

The search is terminated if "--" is found right after the boundary string.

Parameters:

body a placer containing the body text to be parsed.

boundary a pointer to a managed string containing the boundary string for the MIME content.

Returns:

0 on failure, or the number of times the boundary string was located in the MIME body on success.

Definition at line 698 of file mime.c.

References length, log_pedantic, mm_cmp_cs_eq(), pl_empty(), st_char_get(), st_empty, and st_length_get().

Referenced by mail_mime_split().

5.197.2.51 stringer_t* mail_mime_encode_part (stringer_t * data, stringer_t * filename, stringer_t * boundary)

Encode a MIME part for a provided block of data (file attachment) with the specified filename.

Note:

This function will look up the media type based on the supplied filename, and use that media type as a determination of whether the content is to be encoded as either quoted-printable or base64.

Parameters:

data a pointer to a managed string containing the body of the data to be encoded.

filename a pointer to a managed string containing the filename of the attachment for which the data was provided.

boundary a pointer to a managed string containing the boundary that will be used to separate the individual MIME parts.

Returns:

NULL on failure, or a pointer to a managed string containing the file attachment encoded as a MIME part on success.

Definition at line 1082 of file mime.c.

References `base64_encode()`, `media_type_t::bin`, `log_pedantic`, `mail_mime_get_media_type()`, `media_type_t::name`, `qp_encode()`, `st_empty_out()`, `st_free()`, and `st_merge`.

Referenced by `portal_smtp_create_data()`.

5.197.2.52 int_t mail_mime_encoding (placer_t header)

Get the encoding type from a MIME header.

Note:

If no encoding type is specified in the header via Content-Transfer-Encoding, 7bit encoding is assumed.

Parameters:

header a placer containing the MIME header to be examined.

Returns:

the MIME encoding type specified by the header, or `MESSAGE_ENCODING_UNKNOWN` on failure.

Definition at line 574 of file mime.c.

References `mail_header_fetch_cleaned()`, `MESSAGE_ENCODING_7BIT`, `MESSAGE_ENCODING_8BIT`, `MESSAGE_ENCODING_BASE64`, `MESSAGE_ENCODING_QUOTED_PRINTABLE`, `MESSAGE_ENCODING_UNKNOWN`, `mm_cmp_ci_eq()`, `PLACER`, `st_char_get()`, `st_free()`, and `st_length_get()`.

Referenced by `mail_mime_part()`.

5.197.2.53 void mail_mime_free (mail_mime_t * mime)

Free a mail mime object and its underlying data, and recursively free its children parts.

Parameters:

mime a pointer to the mail mime object to be freed.

Returns:

This function returns no value.

Definition at line 888 of file mime.c.

References `ar_field_ptr()`, `ar_free()`, `ar_length_get()`, `mail_mime_t::boundary`, `mail_mime_t::children`, `mail_mime_free()`, `mm_free()`, and `st_cleanup`.

Referenced by `mail_destroy()`, `mail_mime_free()`, `mail_mime_part()`, and `mail_mime_update()`.

5.197.2.54 stringer_t* mail_mime_generate_boundary (array_t * *parts*)

Generate a MIME boundary string that is unique to a collection of content.

Parameters:

parts a pointer to an array of managed strings containing the MIME children data to be separated by the boundary.

Returns:

NULL on failure, or a pointer to a managed string containing the generated boundary on success.

Definition at line 1015 of file mime.c.

References ar_field_ptr(), ar_length_get(), log_error, log_pedantic, rand_get_uint64(), st_alloc(), st_char_get(), st_free(), st_length_set(), and st_search_ci().

Referenced by portal_smtp_create_data().

5.197.2.55 media_type_t* mail_mime_get_media_type (chr_t * *extension*)

Get the media type for a given file extension.

Note:

If no direct match is found for the content, "application/octet-stream" will be returned.

Parameters:

extension a pointer to a null-terminated string containing the file extension to be looked up, starting with a period.

Returns:

a pointer to a media type object corresponding to the media type of the specified file extension.

Definition at line 100 of file mime.c.

References mm_cmp_cs_eq(), ns_empty(), and ns_length_get().

Referenced by mail_mime_encode_part().

5.197.2.56 stringer_t* mail_mime_get_smtp_envelope (stringer_t * *from*, inx_t * *tos*, inx_t * *ccs*, inx_t * *bccs*, stringer_t * *subject*, stringer_t * *boundary*, bool_t *attached*)

Get smtp envelope data for an outbound message sent by a webmail client.

Parameters:

from a pointer to a managed string containing the sender's address.

tos a pointer to an inx holder containing a collection of managed strings with the email addresses specified in the To: header.

ccs a pointer to an inx holder containing a collection of managed strings with the email addresses specified in the CC: header.

bccs a pointer to an inx holder containing a collection of managed strings with the email addresses specified in the BCC: header.

subject a pointer to a managed string containing the text of the subject line.

boundary a pointer to a managed string containing a boundary string to be used in multipart messages.

attached if true, the envelope is to be created for a mail with attachments (type "multipart/mixed"); if false, only a single part will be sent for the main email body.

Returns:

NULL on failure or a pointer to a managed string containing the smtp envelope data that will be supplied to an smtp relay server at the beginning of the DATA command.

Definition at line 1152 of file mime.c.

References `log_error`, `log_pedantic`, `NULLER`, `portal_smtp_merge_headers()`, `st_cleanup`, `st_free()`, and `st_merge`.

Referenced by `portal_smtp_create_data()`.

5.197.2.57 `placer_t mail_mime_header (stringer_t * part)`

Get a placer pointing to a mime header in a mime part.

Parameters:

part a managed string containing the mime part text to be parsed.

Returns:

a placer pointing to the mime header at the start of the specified mime part.

Definition at line 479 of file mime.c.

References `length`, `pl_init()`, `st_char_get()`, `st_data_get()`, and `st_length_get()`.

Referenced by `mail_mime_part()`.

5.197.2.58 `mail_mime_t* mail_mime_part (stringer_t * part, uint32_t recursion)`

Parse a block of data into a mail mime object.

Note:

By parsing the specified mime part, this function fills in the content type and encoding of the resulting mail mime object. If the message is multipart, the boundary string is determined and then used to split the body into children; then each child part is passed to [mail_mime_part\(\)](#) to be parsed likewise, recursively.

Parameters:

part a managed string containing the mime part data to be parsed.

recursion an incremented recursion level tracker for calling this function, to prevent an overflow from occurring.

Returns:

NULL on failure or a pointer to a newly allocated and updated mail mime object parsed from the part data on success.

Definition at line 921 of file mime.c.

References `ar_alloc()`, `ar_append()`, `ar_field_st()`, `ar_free()`, `ar_length_get()`, `ARRAY_TYPE_POINTER`, `mail_mime_t::body`, `mail_mime_t::boundary`, `mail_mime_t::children`, `mail_mime_t::encoding`, `mail_mime_t::entire`, `mail_mime_t::header`, `log_pedantic`, `mail_mime_boundary()`, `mail_mime_encoding()`, `mail_mime_free()`, `mail_mime_header()`, `mail_mime_part()`, `MAIL_MIME_RECURSION_LIMIT`, `mail_mime_split()`, `mail_mime_type()`, `MESSAGE_TYPE_MULTI_ALTERNATIVE`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_MULTI_RELATED`, `MESSAGE_TYPE_MULTI_RFC822`, `MESSAGE_TYPE_MULTI_UNKOWN`, `mm_alloc()`, `pl_init()`, `st_char_get()`, `st_data_get()`, `st_empty`, `st_length_get()`, and `mail_mime_t::type`.

Referenced by `mail_mime_part()`, and `mail_mime_update()`.

5.197.2.59 array_t* mail_mime_split (placer_t *body*, stringer_t * *boundary*)

Split a mime body into an array of children by a boundary string.

Parameters:

body a placer containing the body text to be parsed.

boundary a pointer to a managed string containing the boundary string to split the mime content.

Returns:

NULL on failure, or a pointer to an array of mime children on success.

TODO: This is ugly. Because the array gets a placer pointer we need to free it when done. But that means differentiating between these placers and what were usually passed which will likely stack allocated placers. We could probably just change it to a stringer now that its going to [st_free\(\)](#), but that would mean lots of updates all over the place.

Definition at line 845 of file mime.c.

References [ar_alloc\(\)](#), [ar_append\(\)](#), [ARRAY_TYPE_STRINGER](#), [FOREIGNDATA](#), [HEAP](#), [JOINTED](#), [log_pedantic](#), [mail_mime_child\(\)](#), [mail_mime_count\(\)](#), [pl_set\(\)](#), [placer_t](#), [PLACER_T](#), [st_alloc_opts\(\)](#), [st_empty](#), and [st_free\(\)](#).

Referenced by [mail_mime_part\(\)](#).

5.197.2.60 int_t mail_mime_type (placer_t *header*)

Get the content type from a MIME header.

Note:

If no content type is specified in the header via Content-Type, "text/plain" is assumed.

Parameters:

header a placer containing the MIME header to be examined.

Returns:

the MIME content type specified by the header, or MESSAGE_TYPE_UNKNOWN on failure.

Definition at line 518 of file mime.c.

References [mail_header_fetch_cleaned\(\)](#), [MESSAGE_TYPE_HTML](#), [MESSAGE_TYPE_MULTI_ALTERNATIVE](#), [MESSAGE_TYPE_MULTI_MIXED](#), [MESSAGE_TYPE_MULTI_RELATED](#), [MESSAGE_TYPE_MULTI_RFC822](#), [MESSAGE_TYPE_MULTI_UNKOWN](#), [MESSAGE_TYPE_PLAIN](#), [MESSAGE_TYPE_UNKNOWN](#), [mm_cmp_ci_eq\(\)](#), [PLACER](#), [st_char_get\(\)](#), [st_free\(\)](#), and [st_length_get\(\)](#).

Referenced by [mail_mime_part\(\)](#).

5.197.2.61 stringer_t* mail_mime_type_group (placer_t *header*)

Get the value of the Content-Type header from a mime header.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

a managed string containing the content type value of the header, with "text" as the default.

Definition at line 125 of file mime.c.

References [mail_header_fetch_cleaned\(\)](#), [PLACER](#), [st_char_get\(\)](#), [st_free\(\)](#), [st_import\(\)](#), and [st_length_get\(\)](#).

Referenced by [imap_fetch_bodystructure\(\)](#), and [portal_message_attachments\(\)](#).

5.197.2.62 `array_t* mail_mime_type_parameters (placer_t header)`

Get an array of the key/value pairs of parameters passed to the value of the mime Content-Type header.

Note:

The parameters of the Content-Type header value will be examined, and each found parameter will result in the addition of TWO managed strings to the returned array: the first containing the parameter key name, and the second with the parameter value.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

NULL on failure, or on success, an array of managed strings structured as the key name followed by the value of each parameter passed in the Content-Type header.

Definition at line 431 of file mime.c.

References `ar_alloc()`, `ar_append()`, `ar_free()`, `ar_length_get()`, `ARRAY_TYPE_STRINGER`, `mail_header_fetch_cleaned()`, `mail_mime_type_parameters_key()`, `mail_mime_type_parameters_value()`, `PLACER`, `placer_t`, `st_free()`, `tok_get_count_st()`, `tok_get_st()`, and `upper_st()`.

Referenced by `imap_fetch_bodystructure()`.

5.197.2.63 `stringer_t* mail_mime_type_parameters_key (stringer_t * parameter)`

Get the key name of a mime header line parameter.

Parameters:

parameter a managed string containing the complete parameter (key/value pair) of the mime header line.

Returns:

NULL on failure or a managed string containing the mime header parameter key name on success.

Definition at line 325 of file mime.c.

References `length`, `st_char_get()`, `st_import()`, and `st_length_get()`.

Referenced by `mail_mime_type_parameters()`.

5.197.2.64 `stringer_t* mail_mime_type_parameters_value (stringer_t * parameter)`

Get the value of a mime header line parameter.

Parameters:

parameter a managed string containing the complete parameter (key/value pair) of the mime header line.

Returns:

NULL on failure or a managed string containing the mime header parameter value on success.

Definition at line 363 of file mime.c.

References `length`, `st_char_get()`, `st_import()`, and `st_length_get()`.

Referenced by `mail_mime_type_parameters()`.

5.197.2.65 stringer_t* mail_mime_type_sub (placer_t header)

Get the subtype of the Content-Type header value from a mime header.

Note:

For example in the case of a Content-Type of 'text/plain', "plain" would be the subtype.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

a managed string containing the content subtype of the header, with "plain" as the default.

Definition at line 171 of file mime.c.

References mail_header_fetch_cleaned(), PLACER, st_char_get(), st_free(), st_import(), and st_length_get().

Referenced by imap_fetch_bodystructure(), and portal_message_attachments().

5.197.2.66 int_t mail_mime_update (mail_message_t * message)

Re-parse a mail message's data as a mime part, freeing any existing mime part(s) that may have already existed.

Parameters:

message the mail message object containing the message data to be parsed.

Returns:

This function always returns 1.

Definition at line 996 of file mime.c.

References mail_mime_free(), mail_mime_part(), mail_message_t::mime, pl_init(), placer_t, st_char_get(), st_length_get(), and mail_message_t::text.

Referenced by imap_fetch_message(), imap_fetch_return_message(), imap_fetch_return_mime(), imap_search_messages_body(), imap_search_messages_text(), and portal_endpoint_messages_load().

5.197.2.67 void mail_mod_subject (stringer_t ** message, chr_t * label)

Prepend text to a Subject header line inside of a mail message.

Note:

If no Subject line is found in the header of the message, one will be created and inserted with the specified label at the end of the header.

Parameters:

message a pointer to the address of a managed string that contains the mail message header or mail message body, and will receive the value of the resulting managed string that will be allocated to hold the modified contents.

label a null-terminated string that will be prepended to the value of the subject header value.

Returns:

This function returns no value.

Definition at line 389 of file headers.c.

References CONSTANT, log_pedantic, mail_header_end(), mail_header_pop(), pl_empty(), PLACER, placer_t, st_char_get(), st_cmp_ci_starts(), st_data_get(), st_free(), st_length_get(), and st_merge.

Referenced by mail_add_forward_headers(), mail_load_message(), and smtp_check_filters().

5.197.2.68 `int_t mail_modify_part(server_t *server, mail_message_t *message, stringer_t *part, uint64_t signum, uint64_t sigkey, int_t disposition, int_t recursion)`

Insert a spam signature training link into a specified part of a mime message.

Warning:

This function may fail to work as expected and still return a value of 1.

Note:

If the mime type is MESSAGE_TYPE_UNKNOWN (binary file), the function returns immediately. If the type is MESSAGE_TYPE_HTML or MESSAGE_TYPE_PLAIN, the training link is inserted normally. For MESSAGE_TYPE_MULTI_RELATED, MESSAGE_TYPE_MULTI_MIXED, MESSAGE_TYPE_MULTI_UNKOWN the link is inserted in the first mime part. For MESSAGE_TYPE_MULTI_ALTERNATIVE, the link is inserted into each part, for up to 8 times.

Parameters:

server the server object of the web server where the teacher application is hosted.
message the mail message object of the message to be modified.
part a placer pointing to the specified part of the message to be modified.
signum the spam signature number referenced by the teacher url.
sigkey the spam signature key for client verification in the teacher app.
disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".
recursion an incremented recursion level tracker for calling this function, to prevent an overflow from occurring.

Returns:

0 on recursion failure, or 1 otherwise.

Definition at line 729 of file signatures.c.

References length, log_pedantic, mail_discover_encoding(), mail_discover_type(), mail_get_boundary(), mail_get_chunk(), mail_insert_chunk_base64(), mail_insert_chunk_text(), mail_modify_part(), MAIL_SIGNATURES_RECURSION_LIMIT, MESSAGE_ENCODING_BASE64, MESSAGE_TYPE_HTML, MESSAGE_TYPE_MULTI_ALTERNATIVE, MESSAGE_TYPE_MULTI_MIXED, MESSAGE_TYPE_MULTI_RELATED, MESSAGE_TYPE_MULTI_UNKOWN, MESSAGE_TYPE_PLAIN, MESSAGE_TYPE_UNKNOWN, pl_null(), PLACER, placer_t, st_char_get(), st_data_get(), st_free(), st_length_get(), st_populated, mail_message_t::text, and type().

Referenced by mail_modify_part(), and mail_signature_add().

5.197.2.69 `int_t mail_move_message(uint64_t usernum, uint64_t messagenum, uint64_t source, uint64_t target)`

Move a message to a new folder in the database.

Parameters:

usernum the numerical id of the user that owns the message.
messagenum the numerical id of the message to be moved.
source the numerical id of the current parent folder of the specified message.
target the numerical id of the folder to which the specified message will be moved.

Returns:

-1 on error, 0 if the message wasn't found, or 1 on success.

Definition at line 280 of file `store_message.c`.

References `log_error`, `log_pedantic`, `mail_db_update_message_folder()`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

Referenced by `meta_messages_mover()`.

5.197.2.70 int_t mail_path_finder (chr_t * string)**Note:**

This function is not called from anywhere in the code and may be subject to removal.

Definition at line 13 of file `paths.c`.

References `length`, and `ns_length_get()`.

5.197.2.71 bool_t mail_remove_message (uint64_t usernum, uint64_t messagenum, uint32_t size, chr_t * server)

[remove_message.c](#) [remove_message.c](#)

LOW: When the reliable job queue is working we can handle unlink errors by creating a job entry which will retry the unlink operation at a later time.

Parameters:

usenum the user id to whom the specified mail message belongs.

messagenum the target mail message id.

size the size of the message in bytes, to be assessed against the user quota.

server the name of the server on which the mail message resides.

Returns:

true if the message removal succeeds or false on failure.

Definition at line 20 of file `remove_message.c`.

References `log_pedantic`, `mail_db_delete_message()`, `mail_message_path()`, `ns_free()`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

Referenced by `imap_folder_remove()`, `imap_message_expunge()`, `pop_session_destroy()`, `portal_endpoint_messages_remove()`, and `smtp_rollout()`.

5.197.2.72 void mail_signature_add (mail_message_t * message, server_t * server, uint64_t signum, uint64_t sigkey, int_t disposition)

Insert a spam signature training link into a mail message.

See also:

[mail_modify_part\(\)](#)

Parameters:

message the mail message object of the message to be modified.

server the server object of the web server where the teacher application is hosted.

signum the spam signature number referenced by the teacher url.

sigkey the spam signature key for client verification in the teacher app.

disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".

Returns:

This function returns no value.

Definition at line 849 of file signatures.c.

References log_pedantic, mail_modify_part(), PLACER, st_char_get(), st_length_get(), st_length_int(), and mail_message_t::text.

Referenced by mail_add_forward_headers(), and mail_load_message().

5.197.2.73 **placer_t mail_store_header (chr_t * stream, size_t length)**

Return a placer pointing to the trimmed value of a mail header.

Note:

This function merely marks a string residing between leading whitespace and a trailing or

.

Parameters:

stream a pointer to a buffer containing the start of the mail header data.

length the length, in bytes, of the mail header field. a placer containing the trimmed value of the mail header line.

Definition at line 291 of file headers.c.

References pl_init().

Referenced by mail_headers(), and mail_message().

5.197.2.74 **uint64_t mail_store_message (uint64_t usernum, prime_t * signet, uint64_t foldernum, uint32_t * status, uint64_t signum, uint64_t sigkey, stringer_t * message)**

Store a mail message, with its meta-information in the database, and the contents persisted to disk.

Note:

The stored message is always compressed, but only encrypted if the user's public key is supplied.

Parameters:

usernum the numerical id of the user to which the message belongs.

pubkey if not NULL, a public key that will be used to encrypt the message for the intended user.

foldernum the folder # that will contain the message.

status a pointer to the status flags value for the message, which will be updated if the message is to be encrypted.

signum the spam signature for the message.

sigkey the spam key for the message.

message a managed string containing the raw body of the message.

Returns:

0 on failure, or the newly inserted id of the message in the database on success.

Definition at line 103 of file store_message.c.

References compress_cleanup(), compress_lzo(), compress_total_length(), FMESSAGE_OPT_COMPRESSED, FMESSAGE_OPT_ENCRYPTED, log_error, log_pedantic, mail_db_insert_message(), MAIL_STATUS_ENCRYPTED, mail_store_message_data(), ns_free(), org_key, PLACER, prime_cleanup(), prime_message_encrypt(), st_cleanup, st_length_int(), tran_commit(), tran_rollback(), and tran_start().

Referenced by imap_append_message(), and smtp_store_message().

5.197.2.75 `bool_t mail_store_message_data (uint64_t messagenum, uint8_t flags, stringer_t * data, chr_t ** pathptr)`

Persist a message's data to disk.

Parameters:

messagenum the numerical id of the message that will be associated with the data.

data a pointer to a buffer containing the message's data.

flags the status flags to be stored in the message's on-disk file header.

data_len the size, in bytes, of the message's data.

pathptr if not NULL, the address of a pointer to a string that will receive a copy of the message's on-disk data.

Returns:

true if the storage operation succeeded, or false on failure.

Definition at line 19 of file store_message.c.

References FMESSAGE_MAGIC_1, FMESSAGE_MAGIC_2, log_error, mail_create_directory(), mail_message_path(), message_header_t, ns_free(), st_data_get(), and st_length_get().

Referenced by mail_store_message().

5.198 src/objects/mail/mime.c File Reference

```
#include "magma.h"
```

Functions

- [media_type_t](#) * [mail_mime_get_media_type](#) ([chr_t](#) *extension)
Get the media type for a given file extension.
- [stringer_t](#) * [mail_mime_type_group](#) ([placer_t](#) header)
Get the value of the Content-Type header from a mime header.
- [stringer_t](#) * [mail_mime_type_sub](#) ([placer_t](#) header)
Get the subtype of the Content-Type header value from a mime header.
- [stringer_t](#) * [mail_mime_content_encoding](#) ([placer_t](#) header)
Get the content encoding type from a mime header.
- [stringer_t](#) * [mail_mime_content_id](#) ([placer_t](#) header)
Get the content id from a mime header.
- [stringer_t](#) * [mail_mime_type_parameters_key](#) ([stringer_t](#) *parameter)
Get the key name of a mime header line parameter.
- [stringer_t](#) * [mail_mime_type_parameters_value](#) ([stringer_t](#) *parameter)
Get the value of a mime header line parameter.
- [array_t](#) * [mail_mime_type_parameters](#) ([placer_t](#) header)
Get an array of the key/value pairs of parameters passed to the value of the mime Content-Type header.
- [placer_t](#) [mail_mime_header](#) ([stringer_t](#) *part)
Get a placer pointing to a mime header in a mime part.
- [int_t](#) [mail_mime_type](#) ([placer_t](#) header)
Get the content type from a MIME header.
- [int_t](#) [mail_mime_encoding](#) ([placer_t](#) header)
Get the encoding type from a MIME header.
- [stringer_t](#) * [mail_mime_boundary](#) ([placer_t](#) header)
Get the boundary from a MIME header.
- [uint32_t](#) [mail_mime_count](#) ([placer_t](#) body, [stringer_t](#) *boundary)
Count the number of instances of a boundary string inside a MIME body.
- [placer_t](#) [mail_mime_child](#) ([placer_t](#) body, [stringer_t](#) *boundary, [uint32_t](#) child)
Get a placer pointing to the specified child inside a MIME body.
- [array_t](#) * [mail_mime_split](#) ([placer_t](#) body, [stringer_t](#) *boundary)
Split a mime body into an array of children by a boundary string.
- void [mail_mime_free](#) ([mail_mime_t](#) *mime)

Free a mail mime object and its underlying data, and recursively free its children parts.

- [mail_mime_t * mail_mime_part](#) ([stringer_t](#) *part, uint32_t recursion)

Parse a block of data into a mail mime object.

- [int_t mail_mime_update](#) ([mail_message_t](#) *message)

Re-parse a mail message's data as a mime part, freeing any existing mime part(s) that may have already existed.

- [stringer_t * mail_mime_generate_boundary](#) ([array_t](#) *parts)

Generate a MIME boundary string that is unique to a collection of content.

- [stringer_t * mail_mime_encode_part](#) ([stringer_t](#) *data, [stringer_t](#) *filename, [stringer_t](#) *boundary)

Encode a MIME part for a provided block of data (file attachment) with the specified filename.

- [stringer_t * mail_mime_get_smtp_envelope](#) ([stringer_t](#) *from, [inx_t](#) *tos, [inx_t](#) *ccs, [inx_t](#) *bccs, [stringer_t](#) *subject, [stringer_t](#) *boundary, [bool_t](#) attached)

Get smtp envelope data for an outbound message sent by a webmail client.

Variables

- [media_type_t media_types](#) []

5.198.1 Function Documentation

5.198.1.1 [stringer_t* mail_mime_boundary](#) ([placer_t](#) header)

Get the boundary from a MIME header. [mime.c](#)

Note:

This function first scans the value of Content-Type for the boundary, and then the rest of the MIME header.

Parameters:

header a placer pointing to the MIME header to be parsed.

Returns:

NULL on failure, or a pointer to a managed string containing the MIME boundary string on success.

Definition at line 621 of file mime.c.

References [content](#), [length](#), [log_error](#), [log_pedantic](#), [mail_header_fetch_all\(\)](#), [PLACER](#), [st_char_get\(\)](#), [st_cleanup](#), [st_length_get\(\)](#), [st_merge](#), and [st_search_ci\(\)](#).

Referenced by [mail_mime_part\(\)](#).

5.198.1.2 [placer_t mail_mime_child](#) ([placer_t](#) body, [stringer_t](#) * boundary, uint32_t child)

Get a placer pointing to the specified child inside a MIME body.

Parameters:

body a placer containing the body text to be parsed.

boundary a pointer to a managed string containing the boundary string to split the MIME content.

child the zero-based index of the MIME child to be located in the body text.

Returns:

[pl_null\(\)](#) on failure, or a placer containing the specified MIME child on success.

Definition at line 753 of file mime.c.

References [length](#), [log_pedantic](#), [mm_cmp_cs_eq\(\)](#), [pl_empty\(\)](#), [pl_init\(\)](#), [pl_null\(\)](#), [st_char_get\(\)](#), [st_empty](#), and [st_length_get\(\)](#).

Referenced by [mail_mime_split\(\)](#).

5.198.1.3 stringer_t* mail_mime_content_encoding (placer_t header)

Get the content encoding type from a mime header.

Note:

This function parses the Content-Transfer-Encoding field of the header, returning "7bit" by default.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

a managed string containing the content encoding type value of the header, with "7bit" as default.

Definition at line 235 of file mime.c.

References [mail_header_fetch_cleaned\(\)](#), [PLACER](#), [st_char_get\(\)](#), [st_free\(\)](#), [st_import\(\)](#), and [st_length_get\(\)](#).

Referenced by [imap_fetch_bodystructure\(\)](#).

5.198.1.4 stringer_t* mail_mime_content_id (placer_t header)

Get the content id from a mime header.

Note:

This function parses the Content-Id field of the header, returning "7bit" by default.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

NULL on failure, or a managed string containing the content id value of the header.

Definition at line 281 of file mime.c.

References [mail_header_fetch_cleaned\(\)](#), [PLACER](#), [st_char_get\(\)](#), [st_free\(\)](#), [st_import\(\)](#), and [st_length_get\(\)](#).

Referenced by [imap_fetch_bodystructure\(\)](#).

5.198.1.5 uint32_t mail_mime_count (placer_t body, stringer_t * boundary)

Count the number of instances of a boundary string inside a MIME body.

Note:

The search is terminated if "--" is found right after the boundary string.

Parameters:

body a placer containing the body text to be parsed.

boundary a pointer to a managed string containing the boundary string for the MIME content.

Returns:

0 on failure, or the number of times the boundary string was located in the MIME body on success.

Definition at line 698 of file mime.c.

References `length`, `log_pedantic`, `mm_cmp_cs_eq()`, `pl_empty()`, `st_char_get()`, `st_empty`, and `st_length_get()`.

Referenced by `mail_mime_split()`.

5.198.1.6 stringer_t* mail_mime_encode_part (stringer_t * data, stringer_t * filename, stringer_t * boundary)

Encode a MIME part for a provided block of data (file attachment) with the specified filename.

Note:

This function will look up the media type based on the supplied filename, and use that media type as a determination of whether the content is to be encoded as either quoted-printable or base64.

Parameters:

data a pointer to a managed string containing the body of the data to be encoded.

filename a pointer to a managed string containing the filename of the attachment for which the data was provided.

boundary a pointer to a managed string containing the boundary that will be used to separate the individual MIME parts.

Returns:

NULL on failure, or a pointer to a managed string containing the file attachment encoded as a MIME part on success.

Definition at line 1082 of file mime.c.

References `base64_encode()`, `media_type_t::bin`, `log_pedantic`, `mail_mime_get_media_type()`, `media_type_t::name`, `qp_encode()`, `st_empty_out()`, `st_free()`, and `st_merge`.

Referenced by `portal_smtp_create_data()`.

5.198.1.7 int_t mail_mime_encoding (placer_t header)

Get the encoding type from a MIME header.

Note:

If no encoding type is specified in the header via Content-Transfer-Encoding, 7bit encoding is assumed.

Parameters:

header a placer containing the MIME header to be examined.

Returns:

the MIME encoding type specified by the header, or `MESSAGE_ENCODING_UNKNOWN` on failure.

Definition at line 574 of file mime.c.

References mail_header_fetch_cleaned(), MESSAGE_ENCODING_7BIT, MESSAGE_ENCODING_8BIT, MESSAGE_ENCODING_BASE64, MESSAGE_ENCODING_QUOTED_PRINTABLE, MESSAGE_ENCODING_UNKNOWN, mm_cmp_ci_eq(), PLACER, st_char_get(), st_free(), and st_length_get().

Referenced by mail_mime_part().

5.198.1.8 void mail_mime_free (mail_mime_t * *mime*)

Free a mail mime object and its underlying data, and recursively free its children parts.

Parameters:

mime a pointer to the mail mime object to be freed.

Returns:

This function returns no value.

Definition at line 888 of file mime.c.

References ar_field_ptr(), ar_free(), ar_length_get(), mail_mime_t::boundary, mail_mime_t::children, mail_mime_free(), mm_free(), and st_cleanup.

Referenced by mail_destroy(), mail_mime_free(), mail_mime_part(), and mail_mime_update().

5.198.1.9 stringer_t* mail_mime_generate_boundary (array_t * *parts*)

Generate a MIME boundary string that is unique to a collection of content.

Parameters:

parts a pointer to an array of managed strings containing the MIME children data to be separated by the boundary.

Returns:

NULL on failure, or a pointer to a managed string containing the generated boundary on success.

Definition at line 1015 of file mime.c.

References ar_field_ptr(), ar_length_get(), log_error, log_pedantic, rand_get_uint64(), st_alloc(), st_char_get(), st_free(), st_length_set(), and st_search_ci().

Referenced by portal_smtp_create_data().

5.198.1.10 media_type_t* mail_mime_get_media_type (chr_t * *extension*)

Get the media type for a given file extension.

Note:

If no direct match is found for the content, "application/octet-stream" will be returned.

Parameters:

extension a pointer to a null-terminated string containing the file extension to be looked up, starting with a period.

Returns:

a pointer to a media type object corresponding to the media type of the specified file extension.

Definition at line 100 of file mime.c.

References `mm_cmp_cs_eq()`, `ns_empty()`, and `ns_length_get()`.

Referenced by `mail_mime_encode_part()`.

5.198.1.11 `stringer_t* mail_mime_get_smtp_envelope (stringer_t *from, inx_t *tos, inx_t *ccs, inx_t *bccs, stringer_t *subject, stringer_t *boundary, bool_t attached)`

Get smtp envelope data for an outbound message sent by a webmail client.

Parameters:

from a pointer to a managed string containing the sender's address.

tos a pointer to an inx holder containing a collection of managed strings with the email addresses specified in the To: header.

ccs a pointer to an inx holder containing a collection of managed strings with the email addresses specified in the CC: header.

bccs a pointer to an inx holder containing a collection of managed strings with the email addresses specified in the BCC: header.

subject a pointer to a managed string containing the text of the subject line.

boundary a pointer to a managed string containing a boundary string to be used in multipart messages.

attached if true, the envelope is to be created for a mail with attachments (type "multipart/mixed"); if false, only a single part will be sent for the main email body.

Returns:

NULL on failure or a pointer to a managed string containing the smtp envelope data that will be supplied to an smtp relay server at the beginning of the DATA command.

Definition at line 1152 of file mime.c.

References `log_error`, `log_pedantic`, `NULLER`, `portal_smtp_merge_headers()`, `st_cleanup`, `st_free()`, and `st_merge`.

Referenced by `portal_smtp_create_data()`.

5.198.1.12 `placer_t mail_mime_header (stringer_t *part)`

Get a placer pointing to a mime header in a mime part.

Parameters:

part a managed string containing the mime part text to be parsed.

Returns:

a placer pointing to the mime header at the start of the specified mime part.

Definition at line 479 of file mime.c.

References `length`, `pl_init()`, `st_char_get()`, `st_data_get()`, and `st_length_get()`.

Referenced by `mail_mime_part()`.

5.198.1.13 `mail_mime_t* mail_mime_part (stringer_t *part, uint32_t recursion)`

Parse a block of data into a mail mime object.

Note:

By parsing the specified mime part, this function fills in the content type and encoding of the resulting mail mime object. If the message is multipart, the boundary string is determined and then used to split the body into children; then each child part is passed to [mail_mime_part\(\)](#) to be parsed likewise, recursively.

Parameters:

part a managed string containing the mime part data to be parsed.

recursion an incremented recursion level tracker for calling this function, to prevent an overflow from occurring.

Returns:

NULL on failure or a pointer to a newly allocated and updated mail mime object parsed from the part data on success.

Definition at line 921 of file mime.c.

References `ar_alloc()`, `ar_append()`, `ar_field_st()`, `ar_free()`, `ar_length_get()`, `ARRAY_TYPE_POINTER`, `mail_mime_t::body`, `mail_mime_t::boundary`, `mail_mime_t::children`, `mail_mime_t::encoding`, `mail_mime_t::entire`, `mail_mime_t::header`, `log_pedantic`, `mail_mime_boundary()`, `mail_mime_encoding()`, `mail_mime_free()`, `mail_mime_header()`, `mail_mime_part()`, `MAIL_MIME_RECURSION_LIMIT`, `mail_mime_split()`, `mail_mime_type()`, `MESSAGE_TYPE_MULTI_ALTERNATIVE`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_MULTI_RELATED`, `MESSAGE_TYPE_MULTI_RFC822`, `MESSAGE_TYPE_MULTI_UNKOWN`, `mm_alloc()`, `pl_init()`, `st_char_get()`, `st_data_get()`, `st_empty`, `st_length_get()`, and `mail_mime_t::type`.

Referenced by `mail_mime_part()`, and `mail_mime_update()`.

5.198.1.14 array_t* mail_mime_split (placer_t body, stringer_t * boundary)

Split a mime body into an array of children by a boundary string.

Parameters:

body a placer containing the body text to be parsed.

boundary a pointer to a managed string containing the boundary string to split the mime content.

Returns:

NULL on failure, or a pointer to an array of mime children on success.

TODO: This is ugly. Because the array gets a placer pointer we need to free it when done. But that means differentiating between these placers and what were usually passed which will likely stack allocated placers. We could probably just change it to a stringer now that its going to `st_free()`, but that would mean lots of updates all over the place.

Definition at line 845 of file mime.c.

References `ar_alloc()`, `ar_append()`, `ARRAY_TYPE_STRINGER`, `FOREIGNDATA`, `HEAP`, `JOINTED`, `log_pedantic`, `mail_mime_child()`, `mail_mime_count()`, `pl_set()`, `placer_t`, `PLACER_T`, `st_alloc_opts()`, `st_empty`, and `st_free()`.

Referenced by `mail_mime_part()`.

5.198.1.15 int_t mail_mime_type (placer_t header)

Get the content type from a MIME header.

Note:

If no content type is specified in the header via Content-Type, "text/plain" is assumed.

Parameters:

header a placer containing the MIME header to be examined.

Returns:

the MIME content type specified by the header, or `MESSAGE_TYPE_UNKNOWN` on failure.

Definition at line 518 of file mime.c.

References mail_header_fetch_cleaned(), MESSAGE_TYPE_HTML, MESSAGE_TYPE_MULTI_ALTERNATIVE, MESSAGE_TYPE_MULTI_MIXED, MESSAGE_TYPE_MULTI_RELATED, MESSAGE_TYPE_MULTI_RFC822, MESSAGE_TYPE_MULTI_UNKOWN, MESSAGE_TYPE_PLAIN, MESSAGE_TYPE_UNKNOWN, mm_cmp_ci_eq(), PLACER, st_char_get(), st_free(), and st_length_get().

Referenced by mail_mime_part().

5.198.1.16 stringer_t* mail_mime_type_group (placer_t header)

Get the value of the Content-Type header from a mime header.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

a managed string containing the content type value of the header, with "text" as the default.

Definition at line 125 of file mime.c.

References mail_header_fetch_cleaned(), PLACER, st_char_get(), st_free(), st_import(), and st_length_get().

Referenced by imap_fetch_bodystructure(), and portal_message_attachments().

5.198.1.17 array_t* mail_mime_type_parameters (placer_t header)

Get an array of the key/value pairs of parameters passed to the value of the mime Content-Type header.

Note:

The parameters of the Content-Type header value will be examined, and each found parameter will result in the addition of TWO managed strings to the returned array: the first containing the parameter key name, and the second with the parameter value.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

NULL on failure, or on success, an array of managed strings structured as the key name followed by the value of each parameter passed in the Content-Type header.

Definition at line 431 of file mime.c.

References ar_alloc(), ar_append(), ar_free(), ar_length_get(), ARRAY_TYPE_STRINGER, mail_header_fetch_cleaned(), mail_mime_type_parameters_key(), mail_mime_type_parameters_value(), PLACER, placer_t, st_free(), tok_get_count_st(), tok_get_st(), and upper_st().

Referenced by imap_fetch_bodystructure().

5.198.1.18 stringer_t* mail_mime_type_parameters_key (stringer_t * parameter)

Get the key name of a mime header line parameter.

Parameters:

parameter a managed string containing the complete parameter (key/value pair) of the mime header line.

Returns:

NULL on failure or a managed string containing the mime header parameter key name on success.

Definition at line 325 of file mime.c.

References `length`, `st_char_get()`, `st_import()`, and `st_length_get()`.

Referenced by `mail_mime_type_parameters()`.

5.198.1.19 `stringer_t* mail_mime_type_parameters_value (stringer_t * parameter)`

Get the value of a mime header line parameter.

Parameters:

parameter a managed string containing the complete parameter (key/value pair) of the mime header line.

Returns:

NULL on failure or a managed string containing the mime header parameter value on success.

Definition at line 363 of file mime.c.

References `length`, `st_char_get()`, `st_import()`, and `st_length_get()`.

Referenced by `mail_mime_type_parameters()`.

5.198.1.20 `stringer_t* mail_mime_type_sub (placer_t header)`

Get the subtype of the Content-Type header value from a mime header.

Note:

For example in the case of a Content-Type of 'text/plain', "plain" would be the subtype.

Parameters:

header a placer pointing to the mime header to be parsed.

Returns:

a managed string containing the content subtype of the header, with "plain" as the default.

Definition at line 171 of file mime.c.

References `mail_header_fetch_cleaned()`, `PLACER`, `st_char_get()`, `st_free()`, `st_import()`, and `st_length_get()`.

Referenced by `imap_fetch_bodystructure()`, and `portal_message_attachments()`.

5.198.1.21 `int_t mail_mime_update (mail_message_t * message)`

Re-parse a mail message's data as a mime part, freeing any existing mime part(s) that may have already existed.

Parameters:

message the mail message object containing the message data to be parsed.

Returns:

This function always returns 1.

Definition at line 996 of file mime.c.

References `mail_mime_free()`, `mail_mime_part()`, `mail_message_t::mime`, `pl_init()`, `placer_t`, `st_char_get()`, `st_length_get()`, and `mail_message_t::text`.

Referenced by `imap_fetch_message()`, `imap_fetch_return_message()`, `imap_fetch_return_mime()`, `imap_search_messages_body()`, `imap_search_messages_text()`, and `portal_endpoint_messages_load()`.

5.198.2 Variable Documentation

5.198.2.1 `media_type_t media_types[]`

Definition at line 10 of file mime.c.

5.199 src/objects/mail/objects.c File Reference

```
#include "magma.h"
```

Functions

- void [mail_destroy_message](#) ([smtp_message_t](#) *message)
Destroy an smtp message and free all of its underlying data.
- void [mail_destroy](#) ([mail_message_t](#) *message)
Free a mail message and all its underlying data.
- [mail_message_t](#) * [mail_message](#) ([stringer_t](#) *text)
Parse a raw mail data string and return a new mail message object containing the processed message.
- [smtp_message_t](#) * [mail_create_message](#) ([stringer_t](#) *text)
Create an smtp message object out of a buffer of raw data supplied via the smtp DATA command.

5.199.1 Function Documentation

5.199.1.1 [smtp_message_t](#)* [mail_create_message](#) ([stringer_t](#) * *text*)

Create an smtp message object out of a buffer of raw data supplied via the smtp DATA command.

Note:

This function will also generate a random message ID.

Parameters:

text a pointer to a managed string containing the smtp data to be parsed.

Returns:

NULL on failure or a pointer to the newly initialized smtp message object wrapping the data on success.

Definition at line 127 of file objects.c.

References [smtp_message_t::header_length](#), [smtp_message_t::id](#), [log_pedantic](#), [mail_header_end\(\)](#), [mail_headers\(\)](#), [mm_alloc\(\)](#), [mm_free\(\)](#), [rand_choices\(\)](#), [st_free\(\)](#), and [smtp_message_t::text](#).

Referenced by [smtp_data\(\)](#).

5.199.1.2 void [mail_destroy](#) ([mail_message_t](#) * *message*)

Free a mail message and all its underlying data.

Parameters:

message a pointer to the mail message object to be freed.

Returns:

This function returns no value.

Definition at line 32 of file objects.c.

References mail_message_t::from, mail_mime_free(), mail_message_t::mime, mm_free(), st_cleanup, and mail_message_t::text.

Referenced by imap_fetch_body(), imap_fetch_message(), imap_fetch_return_header(), imap_fetch_return_message(), imap_fetch_return_mime(), imap_fetch_return_text(), imap_search_messages(), mail_add_forward_headers(), mail_load_header(), pop_retr(), pop_top(), and portal_endpoint_messages_load().

5.199.1.3 void mail_destroy_message (smtp_message_t * *message*)

Destroy an smtp message and free all of its underlying data.

Parameters:

message a pointer to the smtp message to be destroyed.

Returns:

This function returns no value.

Definition at line 15 of file objects.c.

References smtp_message_t::from, smtp_message_t::id, mm_free(), st_cleanup, and smtp_message_t::text.

Referenced by smtp_data(), smtp_session_destroy(), and smtp_session_reset().

5.199.1.4 mail_message_t* mail_message (stringer_t * *text*)

Parse a raw mail data string and return a new mail message object containing the processed message. objects.c

Parameters:

text a pointer to the managed string containing the raw (uncompressed) mail data to be parsed.

Returns:

NULL on failure or a pointer to a newly allocated mail message object with the message on success.

Definition at line 54 of file objects.c.

References mail_message_t::date, mail_message_t::from, mail_message_t::header_length, length, log_pedantic, mail_header_end(), mail_header_fetch_cleaned(), mail_store_header(), mm_alloc(), mm_cmp_ci_eq(), mm_free(), PLACER, st_char_get(), st_length_get(), mail_message_t::subject, mail_message_t::text, and mail_message_t::to.

Referenced by mail_add_forward_headers(), and mail_load_message().

5.200 src/objects/objects.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t obj_cache_start](#) (void)
Initialize the object cache for all active user objects and web sessions.
- void [obj_cache_stop](#) (void)
Stop the object cache and free all active user and web session objects.
- void [obj_cache_prune](#) (void)
The prune function runs every few minutes and scans the object cache and removes any stale objects it finds.

Variables

- [object_cache_t](#) objects

5.200.1 Function Documentation

5.200.1.1 void obj_cache_prune (void)

The prune function runs every few minutes and scans the object cache and removes any stale objects it finds.

Note:

If the number of entries is over 4,096, then the prune function will remove entries candidates which have been unused more then 5 minutes. If the index holds more than 2,048, entries older than 30 minutes are pruned, otherwise if the index holds fewer than 2,048 entries, only those objects older than 1 hour are removed. Also, note that the precise interval between scans is somewhat random, because the background thread responsible for running the prune function goes to sleep for a random number of seconds.

Definition at line 64 of file objects.c.

References [count](#), [inx_count\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_key_active\(\)](#), [inx_cursor_reset\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [inx_delete\(\)](#), [inx_lock_read\(\)](#), [inx_lock_write\(\)](#), [inx_unlock\(\)](#), [object_cache_t::meta](#), [meta_user_ref_stamp\(\)](#), [meta_user_ref_total\(\)](#), [meta_user_t](#), [sess_ref_stamp\(\)](#), [sess_ref_total\(\)](#), [session_t](#), [object_cache_t::sessions](#), [stats_adjust_by_name\(\)](#), and [stats_set_by_name\(\)](#).

Referenced by [process_maint\(\)](#).

5.200.1.2 bool_t obj_cache_start (void)

Initialize the object cache for all active user objects and web sessions. objects.c

Returns:

true on success or false on failure.

Definition at line 19 of file objects.c.

References [inx_alloc\(\)](#), [log_critical](#), [M_INX_LOCK_MANUAL](#), [M_INX_TREE](#), [object_cache_t::meta](#), [meta_free\(\)](#), [sess_destroy\(\)](#), and [object_cache_t::sessions](#).

Referenced by [process_start\(\)](#).

5.200.1.3 void obj_cache_stop (void)

Stop the object cache and free all active user and web session objects.

Returns:

This function returns no value.

Definition at line 38 of file objects.c.

References `inx_free()`, `object_cache_t::meta`, and `object_cache_t::sessions`.

Referenced by `process_stop()`.

5.200.2 Variable Documentation

5.200.2.1 object_cache_t objects

Initial value:

```
{  
    .meta = NULL,  
    .sessions = NULL  
}
```

Definition at line 10 of file objects.c.

Referenced by `__attribute__()`, `meta_inx_find()`, `meta_inx_remove()`, `sess_create()`, and `sess_get()`.

5.201 src/providers/prime/transposition/binary/objects.c File Reference

```
#include "magma.h"
```

Functions

- [chr_t * prime_object_type](#) (uint16_t type)
- void [prime_object_free](#) ([prime_object_t](#) *object)
- [prime_object_t * prime_object_alloc](#) (uint16_t type, [prime_size_t](#) size, [prime_size_t](#) fields)
objects.c
- size_t [prime_object_size_max](#) (uint16_t type)
- size_t [prime_object_size_min](#) (uint16_t type)

Variables

- [chr_t * prime_types](#) []

5.201.1 Function Documentation

5.201.1.1 [prime_object_t* prime_object_alloc](#) (uint16_t type, prime_size_t size, prime_size_t fields)

objects.c

Definition at line 66 of file objects.c.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), [prime_field_t](#), and [prime_object_t](#).

Referenced by [prime_unpack\(\)](#).

5.201.1.2 void [prime_object_free](#) ([prime_object_t](#) * object)

Definition at line 57 of file objects.c.

References [mm_free\(\)](#).

Referenced by [org_key_set\(\)](#), [org_signet_set\(\)](#), [prime_unpack\(\)](#), [user_key_set\(\)](#), [user_request_set\(\)](#), and [user_signet_set\(\)](#).

5.201.1.3 size_t [prime_object_size_max](#) (uint16_t type)

Definition at line 87 of file objects.c.

References [log_pedantic](#), [PRIME_MAX_3_BYTE](#), [PRIME_MAX_4_BYTE](#), [PRIME_MESSAGE_ENCRYPTED](#), [PRIME_ORG_KEY](#), [PRIME_ORG_KEY_ENCRYPTED](#), [PRIME_ORG_SIGNET](#), [PRIME_USER_KEY](#), [PRIME_USER_KEY_ENCRYPTED](#), [PRIME_USER_SIGNET](#), and [PRIME_USER_SIGNING_REQUEST](#).

Referenced by [prime_header_write\(\)](#).

5.201.1.4 size_t [prime_object_size_min](#) (uint16_t type)

Definition at line 111 of file objects.c.

References [log_pedantic](#), [PRIME_MESSAGE_ENCRYPTED](#), [PRIME_ORG_KEY](#), [PRIME_ORG_KEY_ENCRYPTED](#), [PRIME_ORG_SIGNET](#), [PRIME_USER_KEY](#), [PRIME_USER_KEY_ENCRYPTED](#), [PRIME_USER_SIGNET](#), and [PRIME_USER_SIGNING_REQUEST](#).

Referenced by [prime_header_write\(\)](#).

5.201.1.5 `chr_t* prime_object_type (uint16_t type)`

Definition at line 21 of file objects.c.

References `log_pedantic`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `prime_types`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_header_write()`, and `prime_pem_wrap()`.

5.201.2 Variable Documentation

5.201.2.1 `chr_t* prime_types[]`

Initial value:

```
{
    "ORGANIZATIONAL SIGNET",
    "ORGANIZATIONAL KEY",
    "ENCRYPTED ORGANIZATIONAL KEY",
    "USER SIGNING REQUEST",
    "USER SIGNET",
    "USER KEY",
    "ENCRYPTED USER KEY",
    "ENCRYPTED MESSAGE"
}
```

Definition at line 10 of file objects.c.

Referenced by `prime_object_type()`.

5.202 src/objects/mail/parsing.c File Reference

```
#include "magma.h"
```

Functions

- [placer_t * mail_domain_get](#) ([stringer_t *address](#), [placer_t *output](#))
Get the domain portion of an email address.
- [stringer_t * mail_extract_address](#) ([stringer_t *address](#))
Extract a valid email address from a From: header value.

5.202.1 Function Documentation

5.202.1.1 [placer_t * mail_domain_get](#) ([stringer_t * address](#), [placer_t * output](#))

Get the domain portion of an email address.

Parameters:

- address*** a managed string containing the email address to be parsed.
output a pointer to a placer to receive the domain portion of the email address.

Returns:

NULL on failure or a pointer to the output of the domain extraction on success.

Definition at line 16 of file parsing.c.

References [tok_get_count_st\(\)](#), and [tok_get_st\(\)](#).

Referenced by [mail_add_outbound_headers\(\)](#), [smtp_check_authorized_from\(\)](#), [smtp_fetch_inbound\(\)](#), [smtp_rcpt_to\(\)](#), and [spf_check\(\)](#).

5.202.1.2 [stringer_t * mail_extract_address](#) ([stringer_t * address](#))

Extract a valid email address from a From: header value. [parsing.c](#)

Note:

The preference of this function is first to try to find the email addressed enclosed in <>. No valid email address may be enclosed in () or "".

Parameters:

address a managed string containing the buffer from which the email address will be extracted.

Returns:

NULL on failure or a managed string containing the email address on success.

Definition at line 32 of file parsing.c.

References [comment](#), [length](#), [log_pedantic](#), [st_alloc\(\)](#), [st_char_get\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), and [st_length_set\(\)](#).

Referenced by [smtp_data_outbound\(\)](#).

5.203 src/objects/mail/paths.c File Reference

```
#include "magma.h"
```

Functions

- `int_t mail_path_finder (chr_t *string)`
- `chr_t * mail_message_path (uint64_t number, chr_t *server)`
Return the fully qualified local file path of a stored mail message for a specified message number and server.
- `bool_t mail_create_directory (uint64_t number, chr_t *server)`
Create the on-disk directory structure necessary to hold a given message's file data.

5.203.1 Function Documentation

5.203.1.1 `bool_t mail_create_directory (uint64_t number, chr_t * server)`

Create the on-disk directory structure necessary to hold a given message's file data.

Parameters:

number the mail message id.

server the hostname of the server where the message data resides or if NULL, the default server.

Returns:

true on success or false on failure.

Definition at line 66 of file paths.c.

References magma_t::active, log_error, magma, magma_t::root, st_char_get(), st_length_int(), and magma_t::storage.

Referenced by mail_copy_message(), and mail_store_message_data().

5.203.1.2 `chr_t* mail_message_path (uint64_t number, chr_t * server)`

Return the fully qualified local file path of a stored mail message for a specified message number and server. [paths.c](#)

Parameters:

number the mail message id.

server the hostname of the server where the message data resides or if NULL, the default server.

Returns:

NULL on failure, or a pointer to a null-terminated string containing the absolute file path of the specified message.

Definition at line 35 of file paths.c.

References magma_t::active, log_pedantic, magma, ns_alloc(), ns_free(), magma_t::root, st_char_get(), st_length_int(), and magma_t::storage.

Referenced by mail_copy_message(), mail_load_message(), mail_remove_message(), and mail_store_message_data().

5.203.1.3 int_t mail_path_finder (chr_t * *string*)**Note:**

This function is not called from anywhere in the code and may be subject to removal.

Definition at line 13 of file paths.c.

References `length`, and `ns_length_get()`.

5.204 src/objects/mail/remove_message.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t mail_remove_message](#) (uint64_t usernum, uint64_t messagenum, uint32_t size, [chr_t](#) *server)

Remove a specified mail message from both the database and storage.

5.204.1 Function Documentation

5.204.1.1 bool_t mail_remove_message (uint64_t usernum, uint64_t messagenum, uint32_t size, chr_t * server)

Remove a specified mail message from both the database and storage. [remove_message.c](#)

LOW: When the reliable job queue is working we can handle unlink errors by creating a job entry which will retry the unlink operation at a later time.

Parameters:

usernum the user id to whom the specified mail message belongs.

messagenum the target mail message id.

size the size of the message in bytes, to be assessed against the user quota.

server the name of the server on which the mail message resides.

Returns:

true if the message removal succeeds or false on failure.

Definition at line 20 of file [remove_message.c](#).

References [log_pedantic](#), [mail_db_delete_message\(\)](#), [mail_message_path\(\)](#), [ns_free\(\)](#), [tran_commit\(\)](#), [tran_rollback\(\)](#), and [tran_start\(\)](#).

Referenced by [imap_folder_remove\(\)](#), [imap_message_expunge\(\)](#), [pop_session_destroy\(\)](#), [portal_endpoint_messages_remove\(\)](#), and [smtp_rollout\(\)](#).

5.205 src/objects/mail/signatures.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * mail_build_signature](#) ([server_t](#) *server, [int_t](#) content_type, [int_t](#) content_encoding, [uint64_t](#) signum, [uint64_t](#) sigkey, [int_t](#) disposition)
Build a spam text signature for insertion into a message body.
- [int_t mail_discover_type](#) ([stringer_t](#) *header)
Get the value of the Content-Type header in a mail message header.
- [int_t mail_discover_encoding](#) ([stringer_t](#) *header)
Get the value of the Content-Transfer-Encoding header in a mail message header.
- [stringer_t * mail_extract_tag](#) ([chr_t](#) *stream, [size_t](#) length)
Extract an html tag from a data buffer, searching backwards.
- [size_t mail_discover_insertion_point](#) ([stringer_t](#) *message, [stringer_t](#) *part, [int_t](#) type)
Find the position in a mail message where a custom message can be inserted.
- [stringer_t * mail_insert_chunk_base64](#) ([server_t](#) *server, [stringer_t](#) *message, [stringer_t](#) *part, [uint64_t](#) signum, [uint64_t](#) sigkey, [int_t](#) disposition, [int_t](#) type, [int_t](#) encoding)
Insert a spam signature training link into a base64-encoded message part.
- [stringer_t * mail_insert_chunk_text](#) ([server_t](#) *server, [stringer_t](#) *message, [stringer_t](#) *part, [uint64_t](#) signum, [uint64_t](#) sigkey, [int_t](#) disposition, [int_t](#) type, [int_t](#) encoding)
Insert a spam signature training link into a plain text message part.
- [placer_t mail_get_chunk](#) ([stringer_t](#) *message, [stringer_t](#) *boundary, [int_t](#) chunk, [bool_t](#) *last)
Get a specified chunk (mime part) of a multipart mime message.
- [stringer_t * mail_get_boundary](#) ([stringer_t](#) *header)
Get the boundary string from a message header.
- [int_t mail_modify_part](#) ([server_t](#) *server, [mail_message_t](#) *message, [stringer_t](#) *part, [uint64_t](#) signum, [uint64_t](#) sigkey, [int_t](#) disposition, [int_t](#) recursion)
Insert a spam signature training link into a specified part of a mime message.
- void [mail_signature_add](#) ([mail_message_t](#) *message, [server_t](#) *server, [uint64_t](#) signum, [uint64_t](#) sigkey, [int_t](#) disposition)
Insert a spam signature training link into a mail message.

5.205.1 Function Documentation

5.205.1.1 [stringer_t* mail_build_signature](#) ([server_t](#) * server, [int_t](#) content_type, [int_t](#) content_encoding, [uint64_t](#) signum, [uint64_t](#) sigkey, [int_t](#) disposition)

Build a spam text signature for insertion into a message body. [signatures.c](#)

Parameters:

server the server object of the web server where the teacher application is hosted.

type MESSAGE_TYPE_HTML to generate an html-based signature, or any other value for plain text.

content_encoding if MESSAGE_ENCODING_QUOTED_PRINTABLE, set the encoding type to qp.

signum the spam signature number referenced by the teacher url.

sigkey the spam signature key for client verification in the teacher app.

disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".

Returns:

NULL on failure, or a newly allocated managed string containing the desired mail signature on success.

Definition at line 20 of file signatures.c.

References server_t::domain, length, log_pedantic, MESSAGE_ENCODING_QUOTED_PRINTABLE, MESSAGE_TYPE_HTML, qp_encode(), st_alloc(), st_char_get(), st_cleanup, st_free(), st_length_get(), st_merge, st_sprint(), and uint64_digits().

Referenced by mail_insert_chunk_base64(), and mail_insert_chunk_text().

5.205.1.2 int_t mail_discover_encoding (stringer_t * header)

Get the value of the Content-Transfer-Encoding header in a mail message header.

Note:

Possible return values include MESSAGE_ENCODING_QUOTED_PRINTABLE, MESSAGE_ENCODING_BASE64, MESSAGE_ENCODING_8BIT, and MESSAGE_ENCODING_7BIT.

Parameters:

header a managed string containing the mail message header to be parsed.

Returns:

the MESSAGE_ENCODING code of the mail transfer encoding type, or MESSAGE_ENCODING_7BIT by default.

Definition at line 171 of file signatures.c.

References content, mail_header_fetch_all(), MESSAGE_ENCODING_7BIT, MESSAGE_ENCODING_8BIT, MESSAGE_ENCODING_BASE64, MESSAGE_ENCODING_QUOTED_PRINTABLE, MESSAGE_ENCODING_UNKNOWN, mm_cmp_ci_eq(), PLACER, st_char_get(), st_free(), and st_length_get().

Referenced by mail_modify_part().

5.205.1.3 size_t mail_discover_insertion_point (stringer_t * message, stringer_t * part, int_t type)

Find the position in a mail message where a custom message can be inserted.

Note:

The part parameter is expected to be a placer pointing into the contents of message. If the encoding type is not html, the insertion point is determined to be at the end of the part. If the encoding type is html, the following rules are followed: 1. Scanning backwards from the end of the message, skip trailing whitespace get the next html tag. 2. If that tag is NOT </html> or </body>, insert the signature AFTER it. 3. If that tag IS </html> or </body>, insert the signature right before it closes.

Parameters:

message a managed string containing the mail message body to be parsed.

part a managed string (placer) containing the part of the message where the custom message should be inserted.

type the encoding type of the message (MESSAGE_TYPE_HTML or other).

Returns:

the zero-based index of the position in the specified message where the custom message can be inserted.

Definition at line 294 of file signatures.c.

References `length`, `mail_extract_tag()`, `MESSAGE_TYPE_HTML`, `PLACER`, `st_char_get()`, `st_cmp_ci_eq()`, `st_data_get()`, `st_free()`, and `st_length_get()`.

Referenced by `mail_insert_chunk_base64()`, and `mail_insert_chunk_text()`.

5.205.1.4 int_t mail_discover_type (stringer_t * header)

Get the value of the Content-Type header in a mail message header.

Note:

Possible return values include `MESSAGE_TYPE_MULTI_ALTERNATIVE`, `MESSAGE_TYPE_MULTI_RELATED`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_HTML`, `MESSAGE_TYPE_MULTI_UNKOWN`, and `MESSAGE_TYPE_PLAIN`.

Parameters:

header a managed string containing the mail message header to be parsed.

Returns:

the `MESSAGE_TYPE` code of the mail content type, or `MESSAGE_TYPE_PLAIN` by default.

Definition at line 120 of file signatures.c.

References `content`, `mail_header_fetch_all()`, `MESSAGE_TYPE_HTML`, `MESSAGE_TYPE_MULTI_ALTERNATIVE`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_MULTI_RELATED`, `MESSAGE_TYPE_MULTI_UNKOWN`, `MESSAGE_TYPE_PLAIN`, `mm_cmp_ci_eq()`, `PLACER`, `st_char_get()`, `st_free()`, and `st_length_get()`.

Referenced by `mail_modify_part()`.

5.205.1.5 stringer_t* mail_extract_tag (chr_t * stream, size_t length)

Extract an html tag from a data buffer, searching backwards.

Warning:

Remember that this function takes the end and not the start of a buffer!

Note:

The extracted tag contains the enclosing brackets and only printable characters (no whitespace).

Parameters:

stream the end of a data buffer containing the html data to be parsed.

length the size, in bytes, of the data buffer to be parsed.

Returns:

NULL on failure, or a managed string containing the printable contents of the nearest html tag on success.

Definition at line 218 of file signatures.c.

References `log_pedantic`, `st_alloc()`, `st_char_get()`, `st_free()`, and `st_length_set()`.

Referenced by `mail_discover_insertion_point()`.

5.205.1.6 **stringer_t*** mail_get_boundary (**stringer_t** * *header*)

Get the boundary string from a message header.

Note:

This function works by parsing the Content-Type header if it exists, falling back to the entire header otherwise. The returned boundary string includes a trailing "--".

Parameters:

header a managed string containing the message header.

Returns:

NULL on failure or a managed string containing the boundary string on success.

Definition at line 643 of file signatures.c.

References content, length, log_pedantic, mail_header_fetch_all(), PLACER, st_char_get(), st_cleanup, st_data_get(), st_length_get(), st_merge, and st_search_ci().

Referenced by mail_modify_part().

5.205.1.7 **placer_t** mail_get_chunk (**stringer_t** * *message*, **stringer_t** * *boundary*, **int_t** *chunk*, **bool_t** * *last*)

Get a specified chunk (mime part) of a multipart mime message.

Parameters:

result a pointer to the placer that will contain the result

message a managed string containing the mime message to be parsed.

boundary a managed string containing the boundary used to split the multipart mime message.

chunk the one-index based chunk to be retrieved from the multipart message

Returns:

NULL on failure or a pointer to the placer containing the specified chunk.

Definition at line 543 of file signatures.c.

References length, log_error, log_pedantic, mm_cmp_cs_eq(), pl_init(), pl_null(), PLACER, st_char_get(), st_cmp_cs_eq(), st_length_get(), and st_search_cs().

Referenced by mail_modify_part().

5.205.1.8 **stringer_t*** mail_insert_chunk_base64 (**server_t** * *server*, **stringer_t** * *message*, **stringer_t** * *part*, **uint64_t** *signum*, **uint64_t** *sigkey*, **int_t** *disposition*, **int_t** *type*, **int_t** *encoding*)

Insert a spam signature training link into a base64-encoded message part.

See also:

[mail_discover_insertion_point\(\)](#)

Note:

This is similar to [mail_insert_chunk_text\(\)](#) except the part has to be decoded and then re-encoded after the training link is inserted.

Parameters:

server the server object of the web server where the teacher application is hosted.

message a managed string containing the base64-encoded message body to be parsed.

part a managed string (placer) containing the part of the message where the signature should be inserted.

signum the spam signature number referenced by the teacher url.

sigkey the spam signature key for client verification in the teacher app.

disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".

type the encoding type of the message (MESSAGE_TYPE_HTML or other).

encoding if MESSAGE_ENCODING_QUOTED_PRINTABLE, set the encoding type to qp.

Returns:

NULL on failure or a managed string containing the base64-encoded message with the inserted signature training link on success.

Definition at line 359 of file signatures.c.

References base64_decode(), base64_encode(), length, log_pedantic, mail_build_signature(), mail_discover_insertion_point(), PLACER, st_char_get(), st_cleanup, st_data_get(), st_free(), st_length_get(), and st_merge.

Referenced by mail_modify_part().

5.205.1.9 stringer_t* mail_insert_chunk_text (server_t * server, stringer_t * message, stringer_t * part, uint64_t signum, uint64_t sigkey, int_t disposition, int_t type, int_t encoding)

Insert a spam signature training link into a plain text message part.

See also:

[mail_discover_insertion_point\(\)](#)

Parameters:

server the server object of the web server where the teacher application is hosted.

message a managed string containing the message body to be parsed.

part a managed string (placer) containing the part of the message where the signature should be inserted.

signum the spam signature number referenced by the teacher url.

sigkey the spam signature key for client verification in the teacher app.

disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".

type the encoding type of the message (MESSAGE_TYPE_HTML or other).

encoding if MESSAGE_ENCODING_QUOTED_PRINTABLE, set the encoding type to qp.

Returns:

NULL on failure or a managed string containing the message with the inserted signature training link on success.

Definition at line 500 of file signatures.c.

References log_pedantic, mail_build_signature(), mail_discover_insertion_point(), PLACER, st_char_get(), st_empty, st_free(), st_length_get(), and st_merge.

Referenced by mail_modify_part().

5.205.1.10 `int_t mail_modify_part(server_t * server, mail_message_t * message, stringer_t * part, uint64_t signum, uint64_t sigkey, int_t disposition, int_t recursion)`

Insert a spam signature training link into a specified part of a mime message.

Warning:

This function may fail to work as expected and still return a value of 1.

Note:

If the mime type is MESSAGE_TYPE_UNKNOWN (binary file), the function returns immediately. If the type is MESSAGE_TYPE_HTML or MESSAGE_TYPE_PLAIN, the training link is inserted normally. For MESSAGE_TYPE_MULTI_RELATED, MESSAGE_TYPE_MULTI_MIXED, MESSAGE_TYPE_MULTI_UNKOWN the link is inserted in the first mime part. For MESSAGE_TYPE_MULTI_ALTERNATIVE, the link is inserted into each part, for up to 8 times.

Parameters:

server the server object of the web server where the teacher application is hosted.
message the mail message object of the message to be modified.
part a placer pointing to the specified part of the message to be modified.
signum the spam signature number referenced by the teacher url.
sigkey the spam signature key for client verification in the teacher app.
disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".
recursion an incremented recursion level tracker for calling this function, to prevent an overflow from occurring.

Returns:

0 on recursion failure, or 1 otherwise.

Definition at line 729 of file signatures.c.

References `length`, `log_pedantic`, `mail_discover_encoding()`, `mail_discover_type()`, `mail_get_boundary()`, `mail_get_chunk()`, `mail_insert_chunk_base64()`, `mail_insert_chunk_text()`, `mail_modify_part()`, `MAIL_SIGNATURES_RECURSION_LIMIT`, `MESSAGE_ENCODING_BASE64`, `MESSAGE_TYPE_HTML`, `MESSAGE_TYPE_MULTI_ALTERNATIVE`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_MULTI_RELATED`, `MESSAGE_TYPE_MULTI_UNKOWN`, `MESSAGE_TYPE_PLAIN`, `MESSAGE_TYPE_UNKNOWN`, `pl_null()`, `PLACER`, `placer_t`, `st_char_get()`, `st_data_get()`, `st_free()`, `st_length_get()`, `st_populated`, `mail_message_t::text`, and `type()`.

Referenced by `mail_modify_part()`, and `mail_signature_add()`.

5.205.1.11 `void mail_signature_add(mail_message_t * message, server_t * server, uint64_t signum, uint64_t sigkey, int_t disposition)`

Insert a spam signature training link into a mail message.

See also:

[mail_modify_part\(\)](#)

Parameters:

message the mail message object of the message to be modified.
server the server object of the web server where the teacher application is hosted.
signum the spam signature number referenced by the teacher url.
sigkey the spam signature key for client verification in the teacher app.
disposition if 0, the message disposition is "innocent"; otherwise, the disposition specifies "spam".

Returns:

This function returns no value.

Definition at line 849 of file signatures.c.

References `log_pedantic`, `mail_modify_part()`, `PLACER`, `st_char_get()`, `st_length_get()`, `st_length_int()`, and `mail_message_t::text`.

Referenced by `mail_add_forward_headers()`, and `mail_load_message()`.

5.206 src/objects/mail/store_message.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t mail_store_message_data](#) (uint64_t messagenum, uint8_t flags, [stringer_t](#) *data, [chr_t](#) **pathptr)
Persist a message's data to disk.
- [uint64_t mail_store_message](#) (uint64_t usernum, [prime_t](#) *signet, uint64_t foldernum, uint32_t *status, uint64_t signum, uint64_t sigkey, [stringer_t](#) *message)
Store a mail message, with its meta-information in the database, and the contents persisted to disk.
- [uint64_t mail_copy_message](#) (uint64_t usernum, uint64_t original, [chr_t](#) *server, uint32_t size, uint64_t foldernum, uint32_t status, uint64_t signum, uint64_t sigkey, uint64_t created)
Create a copy of a mail message, with a new entry in the database and a hard link to the message contents on disk.
- [int_t mail_move_message](#) (uint64_t usernum, uint64_t messagenum, uint64_t source, uint64_t target)
Move a message to a new folder in the database.

5.206.1 Function Documentation

5.206.1.1 [uint64_t mail_copy_message](#) (uint64_t *usernum*, uint64_t *original*, [chr_t](#) * *server*, uint32_t *size*, uint64_t *foldernum*, uint32_t *status*, uint64_t *signum*, uint64_t *sigkey*, uint64_t *created*)

Create a copy of a mail message, with a new entry in the database and a hard link to the message contents on disk. [store_message.c](#)

Parameters:

- usernum*** the numerical id of the user to whom the mail message belongs.
- original*** the numerical id of the mail message to be copied.
- server*** a pointer to a null-terminated string containing the name of the server where the message contents are stored.
- size*** the size, in bytes, of the mail message to be copied.
- foldernum*** the numerical id of the folder to become the parent folder of the message copy.
- signum*** the spam signature for the message.
- sigkey*** the spam key for the message.
- created*** the UNIX timestamp of when the message was created.

Returns:

- 0 on failure, or the ID of the copy of the mail message in the database on success.

Definition at line 195 of file store_message.c.

References [log_error](#), [log_pedantic](#), [mail_create_directory\(\)](#), [mail_db_insert_duplicate_message\(\)](#), [mail_message_path\(\)](#), [ns_free\(\)](#), [tran_commit\(\)](#), [tran_rollback\(\)](#), and [tran_start\(\)](#).

Referenced by [imap_message_copier\(\)](#), and [meta_messages_copier\(\)](#).

5.206.1.2 int_t mail_move_message (uint64_t usernum, uint64_t messagenum, uint64_t source, uint64_t target)

Move a message to a new folder in the database.

Parameters:

- usernum* the numerical id of the user that owns the message.
- messagenum* the numerical id of the message to be moved.
- source* the numerical id of the current parent folder of the specified message.
- target* the numerical id of the folder to which the specified message will be moved.

Returns:

- 1 on error, 0 if the message wasn't found, or 1 on success.

Definition at line 280 of file store_message.c.

References log_error, log_pedantic, mail_db_update_message_folder(), tran_commit(), tran_rollback(), and tran_start().

Referenced by meta_messages_mover().

5.206.1.3 uint64_t mail_store_message (uint64_t usernum, prime_t * signet, uint64_t foldernum, uint32_t * status, uint64_t signum, uint64_t sigkey, stringer_t * message)

Store a mail message, with its meta-information in the database, and the contents persisted to disk.

Note:

- The stored message is always compressed, but only encrypted if the user's public key is supplied.

Parameters:

- usernum* the numerical id of the user to which the message belongs.
- pubkey* if not NULL, a public key that will be used to encrypt the message for the intended user.
- foldernum* the folder # that will contain the message.
- status* a pointer to the status flags value for the message, which will be updated if the message is to be encrypted.
- signum* the spam signature for the message.
- sigkey* the spam key for the message.
- message* a managed string containing the raw body of the message.

Returns:

- 0 on failure, or the newly inserted id of the message in the database on success.

Definition at line 103 of file store_message.c.

References compress_cleanup(), compress_lzo(), compress_total_length(), FMESSAGE_OPT_COMPRESSED, FMESSAGE_OPT_ENCRYPTED, log_error, log_pedantic, mail_db_insert_message(), MAIL_STATUS_ENCRYPTED, mail_store_message_data(), ns_free(), org_key, PLACER, prime_cleanup(), prime_message_encrypt(), st_cleanup, st_length_int(), tran_commit(), tran_rollback(), and tran_start().

Referenced by imap_append_message(), and smtp_store_message().

5.206.1.4 bool_t mail_store_message_data (uint64_t messagenum, uint8_t flags, stringer_t * data, chr_t ** pathptr)

Persist a message's data to disk.

Parameters:

messagenum the numerical id of the message that will be associated with the data.

data a pointer to a buffer containing the message's data.

fflags the status flags to be stored in the message's on-disk file header.

data_len the size, in bytes, of the message's data.

pathptr if not NULL, the address of a pointer to a string that will receive a copy of the message's on-disk data.

Returns:

true if the storage operation succeeded, or false on failure.

Definition at line 19 of file store_message.c.

References FMESSAGE_MAGIC_1, FMESSAGE_MAGIC_2, log_error, mail_create_directory(), mail_message_path(), message_header_t, ns_free(), st_data_get(), and st_length_get().

Referenced by mail_store_message().

5.207 src/objects/messages/messages.h File Reference

Defines

- #define [MAIL_STATUS_USER_FLAGS](#) (MAIL_STATUS_SEEN | MAIL_STATUS_ANSWERED | MAIL_STATUS_FLAGGED | MAIL_STATUS_DELETED | MAIL_STATUS_DRAFT)
- #define [MAIL_STATUS_SYSTEM_FLAGS](#)
- #define [MAIL_STATUS_ALL_FLAGS](#) (MAIL_STATUS_USER_FLAGS | MAIL_STATUS_SYSTEM_FLAGS)
- #define [MESSAGE_TYPE_UNKNOWN](#) 0
- #define [MESSAGE_TYPE_PLAIN](#) 1
- #define [MESSAGE_TYPE_HTML](#) 2
- #define [MESSAGE_TYPE_MULTI_ALTERNATIVE](#) 3
- #define [MESSAGE_TYPE_MULTI_MIXED](#) 4
- #define [MESSAGE_TYPE_MULTI_RELATED](#) 5
- #define [MESSAGE_TYPE_MULTI_RFC822](#) 5
- #define [MESSAGE_TYPE_MULTI_UNKOWN](#) 6
- #define [MESSAGE_ENCODING_UNKNOWN](#) 0
- #define [MESSAGE_ENCODING_QUOTED_PRINTABLE](#) 1
- #define [MESSAGE_ENCODING_BASE64](#) 2
- #define [MESSAGE_ENCODING_7BIT](#) 3
- #define [MESSAGE_ENCODING_8BIT](#) 4
- #define [FMESSAGE_MAGIC_1](#) 0x17
- #define [FMESSAGE_MAGIC_2](#) 0x76
- #define [FMESSAGE_OPT_COMPRESSED](#) 0x1
- #define [FMESSAGE_OPT_ENCRYPTED](#) 0x2

Enumerations

- enum {
[MAIL_STATUS_NONE](#) = 0, [MAIL_STATUS_EMPTY](#) = 1, [MAIL_STATUS_RECENT](#) = 2, [MAIL_STATUS_SEEN](#) = 4,
[MAIL_STATUS_ANSWERED](#) = 8, [MAIL_STATUS_FLAGGED](#) = 16, [MAIL_STATUS_DELETED](#) = 32, [MAIL_STATUS_DRAFT](#) = 64,
[MAIL_STATUS_SECURE](#) = 128, [MAIL_STATUS_APPENDED](#) = 256, [MAIL_STATUS_HIDDEN](#) = 512, [MAIL_MARK_JUNK](#) = 1024,
[MAIL_MARK_INFECTED](#) = 2048, [MAIL_MARK_SPOOFED](#) = 4096, [MAIL_MARK_BLACKHOLED](#) = 8192, [MAIL_MARK_PHISHING](#) = 16384,
[MAIL_STATUS_TAGGED](#) = 32768, [MAIL_STATUS_ENCRYPTED](#) = 65536 }

Functions

- struct [__attribute__](#) ((packed))
- [message_t](#) * [message_alloc](#) (uint64_t messagenum, uint64_t created, uint64_t signature, uint64_t key, uint64_t flags, [stringer_t](#) *server, size_t size)
messages.c
- void [message_free](#) ([message_t](#) *message)
Free a message object and its underlying data.
- [inx_t](#) * [messages_update](#) (uint64_t usernum)
Fetch all of a user's message folders and their child messages from the database.

- `meta_message_t * meta_message_by_number (inx_t *messages, uint64_t number)`
meta.c
- `meta_message_t * meta_message_dupe (meta_message_t *message)`
Duplicate a meta message object (along with a deep copy of its tags).
- `void meta_message_free (meta_message_t *message)`
Free a meta message object (and its tags).
- `bool_t meta_messages_copier (meta_user_t *user, meta_message_t *message, uint64_t target, uint64_t *outnum, bool_t sequences, META_LOCK_STATUS locked)`
- `bool_t meta_messages_login_update (meta_user_t *user, META_LOCK_STATUS locked)`
Build a user's messages collection if it is empty, or needs to be refreshed (see note).
- `int_t meta_messages_mover (meta_user_t *user, meta_message_t *message, uint64_t target, bool_t lookup, bool_t sequences, META_LOCK_STATUS locked)`
- `int_t meta_messages_update (meta_user_t *user, META_LOCK_STATUS locked)`
Refresh and resquence a user's message collection if it is stale.
- `void meta_messages_update_sequences (inx_t *folders, inx_t *messages)`
Update the sequence numbers of a series of messages, each re-indexed by their containing folder.
- `bool_t meta_data_fetch_folder_messages (uint64_t usernum, message_folder_t *folder)`
datatier.c
- `void meta_data_fetch_message_tags (meta_message_t *message)`
Fetch the tags for a specified message from the database.
- `bool_t meta_data_fetch_messages (meta_user_t *user)`
Fetch all of a user's stored messages from the database and attach them to the meta user object.

Variables

- `message_t`
- `message_header_t`

5.207.1 Define Documentation

5.207.1.1 #define FMESSAGE_MAGIC_1 0x17

Definition at line 102 of file messages.h.

Referenced by `mail_load_message()`, and `mail_store_message_data()`.

5.207.1.2 #define FMESSAGE_MAGIC_2 0x76

Definition at line 103 of file messages.h.

Referenced by `mail_load_message()`, and `mail_store_message_data()`.

5.207.1.3 #define FMESSAGE_OPT_COMPRESSED 0x1

Definition at line 105 of file messages.h.

Referenced by mail_load_message(), and mail_store_message().

5.207.1.4 #define FMESSAGE_OPT_ENCRYPTED 0x2

Definition at line 106 of file messages.h.

Referenced by mail_load_message(), and mail_store_message().

5.207.1.5 #define MAIL_STATUS_ALL_FLAGS (MAIL_STATUS_USER_FLAGS | MAIL_STATUS_SYSTEM_FLAGS)

Definition at line 45 of file messages.h.

5.207.1.6 #define MAIL_STATUS_SYSTEM_FLAGS**Value:**

```
(MAIL_STATUS_EMPTY | MAIL_STATUS_RECENT | MAIL_STATUS_SECURE |  
MAIL_STATUS_APPENDED | MAIL_STATUS_HIDDEN | \  
MAIL_MARK_JUNK | MAIL_MARK_INFECTED | MAIL_MARK_SPOOFED | MAIL_MARK_BLACKHOLED  
| MAIL_MARK_PHISHING | MAIL_STATUS_TAGGED | MAIL_STATUS_ENCRYPTED)
```

Definition at line 41 of file messages.h.

Referenced by portal_endpoint_messages_flag().

**5.207.1.7 #define MAIL_STATUS_USER_FLAGS (MAIL_STATUS_SEEN | MAIL_STATUS_ANSWERED |
MAIL_STATUS_FLAGGED | MAIL_STATUS_DELETED | MAIL_STATUS_DRAFT)**

Definition at line 38 of file messages.h.

Referenced by meta_data_flags_replace(), and portal_endpoint_messages_flag().

5.207.1.8 #define MESSAGE_ENCODING_7BIT 3

Definition at line 61 of file messages.h.

Referenced by mail_discover_encoding(), and mail_mime_encoding().

5.207.1.9 #define MESSAGE_ENCODING_8BIT 4

Definition at line 62 of file messages.h.

Referenced by mail_discover_encoding(), and mail_mime_encoding().

5.207.1.10 #define MESSAGE_ENCODING_BASE64 2

Definition at line 60 of file messages.h.

Referenced by mail_discover_encoding(), mail_mime_encoding(), and mail_modify_part().

5.207.1.11 #define MESSAGE_ENCODING_QUOTED_PRINTABLE 1

Definition at line 59 of file messages.h.

Referenced by mail_build_signature(), mail_discover_encoding(), and mail_mime_encoding().

5.207.1.12 #define MESSAGE_ENCODING_UNKNOWN 0

Definition at line 58 of file messages.h.

Referenced by mail_discover_encoding(), and mail_mime_encoding().

5.207.1.13 #define MESSAGE_TYPE_HTML 2

Definition at line 50 of file messages.h.

Referenced by mail_build_signature(), mail_discover_insertion_point(), mail_discover_type(), mail_mime_type(), mail_modify_part(), portal_message_attachments(), and portal_message_body().

5.207.1.14 #define MESSAGE_TYPE_MULTI_ALTERNATIVE 3

Definition at line 51 of file messages.h.

Referenced by mail_discover_type(), mail_mime_part(), mail_mime_type(), mail_modify_part(), and portal_message_body().

5.207.1.15 #define MESSAGE_TYPE_MULTI_MIXED 4

Definition at line 52 of file messages.h.

Referenced by mail_discover_type(), mail_mime_part(), mail_mime_type(), mail_modify_part(), portal_message_attachments(), and portal_message_body().

5.207.1.16 #define MESSAGE_TYPE_MULTI_RELATED 5

Definition at line 53 of file messages.h.

Referenced by mail_discover_type(), mail_mime_part(), mail_mime_type(), mail_modify_part(), and portal_message_body().

5.207.1.17 #define MESSAGE_TYPE_MULTI_RFC822 5

Definition at line 54 of file messages.h.

Referenced by mail_mime_part(), and mail_mime_type().

5.207.1.18 #define MESSAGE_TYPE_MULTI_UNKOWN 6

Definition at line 55 of file messages.h.

Referenced by mail_discover_type(), mail_mime_part(), mail_mime_type(), mail_modify_part(), and portal_message_body().

5.207.1.19 #define MESSAGE_TYPE_PLAIN 1

Definition at line 49 of file messages.h.

Referenced by mail_discover_type(), mail_mime_type(), mail_modify_part(), portal_message_attachments(), and portal_message_body().

5.207.1.20 #define MESSAGE_TYPE_UNKNOWN 0

Definition at line 48 of file messages.h.

Referenced by mail_mime_type(), and mail_modify_part().

5.207.2 Enumeration Type Documentation

5.207.2.1 anonymous enum

Enumerator:

MAIL_STATUS_NONE
MAIL_STATUS_EMPTY
MAIL_STATUS_RECENT
MAIL_STATUS_SEEN
MAIL_STATUS_ANSWERED
MAIL_STATUS_FLAGGED
MAIL_STATUS_DELETED
MAIL_STATUS_DRAFT
MAIL_STATUS_SECURE
MAIL_STATUS_APPENDED
MAIL_STATUS_HIDDEN
MAIL_MARK_JUNK
MAIL_MARK_INFECTED
MAIL_MARK_SPOOFED
MAIL_MARK_BLACKHOLED
MAIL_MARK_PHISHING
MAIL_STATUS_TAGGED
MAIL_STATUS_ENCRYPTED

Definition at line 12 of file messages.h.

5.207.3 Function Documentation

5.207.3.1 struct __attribute__((packed)) [read]

LOW: Update the Messages table columns so they match the tank mail message header fields. Store individual header/body lengths and the compressed/uncompressed hash values; then update the message type to store and use the new information.

Definition at line 108 of file messages.h.

5.207.3.2 message_t* message_alloc (uint64_t messagenum, uint64_t created, uint64_t signature, uint64_t key, uint64_t flags, stringer_t * server, size_t size)

messages.c messages.c

Parameters:

messagenum the numerical message id of the new message.

created the message creation timestamp.

signature the message's spam filter signature number.

key the message's spam filter access key.

flags the message's flags value.

server a managed string containing the name of the server where the message will be stored.

size the size, in bytes, of the raw message data.

Returns:

NULL on failure, or the newly allocated message object on success.

Definition at line 60 of file messages.c.

References align(), FOREIGNDATA, JOINTED, log_pedantic, message_t, mm_alloc(), mm_copy(), mm_free(), PLACER_T, placer_t, rwlock_init(), st_data_get(), st_length_get(), and STACK.

Referenced by meta_data_fetch_folder_messages().

5.207.3.3 void message_free (message_t * message)

Free a message object and its underlying data.

Returns:

This function returns no value.

Definition at line 39 of file messages.c.

References mm_cleanup, rwlock_destroy(), and st_cleanup.

Referenced by message_folder_alloc(), and meta_data_fetch_folder_messages().

5.207.3.4 inx_t* messages_update (uint64_t usernum)

Fetch all of a user's message folders and their child messages from the database.

Parameters:

usernum the numerical id of the target user.

Returns:

NULL on failure or an inx object holding all the retrieved and populated message folder objects on success.

Definition at line 96 of file messages.c.

References inx_cleanup(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_free(), inx_t, log_pedantic, M_FOLDER_MESSAGES, magma_folder_fetch(), and meta_data_fetch_folder_messages().

Referenced by meta_update_message_folders().

5.207.3.5 bool_t meta_data_fetch_folder_messages (uint64_t usernum, message_folder_t * folder)

datatier.c datatier.c

Parameters:

usernum the numerical id of the user that owns the folder.

folder a pointer to the message folder object to be populated.

Returns:

true on success or false on failure.

Definition at line 16 of file datatier.c.

References `inx_append()`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `message_alloc()`, `message_free()`, `message_t`, `mm_wipe()`, `PLACER`, `res_field_block()`, `res_field_length()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `messages_update()`.

5.207.3.6 void meta_data_fetch_message_tags (meta_message_t * message)

Fetch the tags for a specified message from the database.

Note:

The results of the operation will be stored in the specified meta message object's "tags" member.

Parameters:

message the meta message object for which the tags will be looked up.

Returns:

This function returns no value.

Definition at line 77 of file datatier.c.

References `ar_alloc()`, `ar_append()`, `ARRAY_TYPE_STRINGER`, `mm_wipe()`, `res_field_string()`, `res_row_count()`, `res_row_next()`, `res_table_free()`, and `stmt_get_result()`.

Referenced by `meta_data_fetch_messages()`, and `portal_endpoint_messages_tag()`.

5.207.3.7 bool_t meta_data_fetch_messages (meta_user_t * user)

Fetch all of a user's stored messages from the database and attach them to the meta user object.

Note:

Any of the user's existing messages will be destroyed first to allow for updates.

Parameters:

user the meta user object whose mail messages will be retrieved.

Returns:

true on success or false on failure.

TODO: Do we still need this once the refactorization is complete?

Definition at line 115 of file datatier.c.

References `inx_alloc()`, `inx_append()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_truncate()`, `log_error`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `MAIL_STATUS_TAGGED`, `meta_data_fetch_message_tags()`, `meta_message_free()`, `meta_message_t`, `mm_alloc()`, `mm_copy()`, `mm_free()`, `mm_wipe()`, `res_field_block()`, `res_field_length()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `stmt_get_result()`, `multi_t::type`, `multi_t::u64`, and `multi_t::val`.

Referenced by `meta_messages_login_update()`, and `meta_messages_update()`.

5.207.3.8 **meta_message_t* meta_message_by_number** (inx_t * *messages*, uint64_t *number*)

meta.c meta.c

Parameters:

messages a pointer to the messages collection to be searched.

number the number of the message to be retrieved.

Returns:

NULL on failure, or a pointer to the meta message object of the matching mail message.

Definition at line 54 of file meta.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, and `meta_message_t`.

5.207.3.9 **meta_message_t* meta_message_dupe** (meta_message_t * *message*)

Duplicate a meta message object (along with a deep copy of its tags).

Parameters:

message the meta message object to be cloned.

Returns:

NULL on failure or a pointer to the duplicated meta message object on success.

Definition at line 33 of file meta.c.

References `ar_dupe()`, `meta_message_t`, and `mm_dupe()`.

Referenced by `imap_duplicate_messages()`, `imap_message_copier()`, `imap_search_messages()`, and `meta_messages_copier()`.

5.207.3.10 **void meta_message_free** (meta_message_t * *message*)

Free a meta message object (and its tags).

Returns:

This function returns no value.

Definition at line 14 of file meta.c.

References `ar_free()`, and `mm_free()`.

Referenced by `imap_duplicate_messages()`, `imap_search_messages()`, and `meta_data_fetch_messages()`.

5.207.3.11 **bool_t meta_messages_copier** (meta_user_t * *user*, meta_message_t * *message*, uint64_t *target*, uint64_t * *outnum*, bool_t *sequences*, META_LOCK_STATUS *locked*)

Parameters:

Make a copy of a mail message, in the database and in memory.

See also:

[mail_copy_message\(\)](#)

Note:

The message copy will not have the deleted, hidden, or recent flags set in the database, but it will have the recent flag set in memory.

Parameters:

user a pointer to the meta user object to whom the message belong.
message the meta message object of the message to be copied.
target the numerical id of the folder to which the message will be copied.
outnum a pointer to an unsigned 64-bit integer that will receive the value of the numerical id of the message copy.
sequences if true, resequence the message folders after the copy has been made.
locked if set to META_NEED_LOCK, lock the specified meta user object for the duration of the request.

Returns:

true on success or false on failure.

Definition at line 238 of file meta.c.

References `inx_insert()`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `mail_copy_message()`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_HIDDEN`, `MAIL_STATUS_RECENT`, `meta_message_dupe()`, `meta_message_t`, `meta_messages_update_sequences()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `mm_free()`, `status`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_endpoint_messages_copy()`.

5.207.3.12 bool_t meta_messages_login_update (meta_user_t * user, META_LOCK_STATUS locked)

Build a user's messages collection if it is empty, or needs to be refreshed (see note).

Note:

This function will fetch and sequence the user's messages from the database if the meta user object has no messages, or if the user has one or fewer pop sessions open and the messages serial number was stale.

Parameters:

user a pointer to the meta user object of the user requesting the messages update.
locked if set to META_NEED_LOCK, lock the specified meta user object for the duration of the request.

Returns:

true if no update was necessary or if the update was successful, or false otherwise.

Definition at line 125 of file meta.c.

References `meta_data_fetch_messages()`, `meta_messages_update_sequences()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_MESSAGES`, `serial_get()`, and `serial_increment()`.

Referenced by `pop_pass()`.

5.207.3.13 int_t meta_messages_mover (meta_user_t * user, meta_message_t * message, uint64_t target, bool_t lookup, bool_t sequences, META_LOCK_STATUS locked)**Parameters:**

Move a mail message to another folder.

See also:

[mail_move_message\(\)](#)

Note:

The message will receive a copy of the recent flag in memory.

Parameters:

user a pointer to the meta user object to whom the message belong.

message the meta message object of the message to be moved.

target the numerical id of the folder to which the message will be copied.

lookup if set, verify the existence of the moved message in the user's messages.

sequences if true, resequence the message folders after the copy has been made.

locked if set to META_NEED_LOCK, lock the specified meta user object for the duration of the request.

Returns:

true on success or false on failure.

BUG: Currently we preserve the original message number which means when the target folder sequences are updated the moved message can appear anywhere in the list (since its based on message number). Inserting messages into the sequence list in this fashion will probably break clients. We may want to duplicate the row which would generate a new message number causing the newly added message to appear at the end of the folder listing. On the other hand POP ignores folders completely, so maybe it won't be quite so bad.

Definition at line 316 of file meta.c.

References `inx_find()`, `log_pedantic`, `M_TYPE_UINT64`, `mail_move_message()`, `MAIL_STATUS_RECENT`, `meta_messages_update_sequences()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_endpoint_messages_move()`.

5.207.3.14 int_t meta_messages_update (meta_user_t * user, META_LOCK_STATUS locked)

Refresh and resquence a user's message collection if it is stale.

Note:

The user's messages will only be updated if they are empty or if they are out of sync and the user has no open pop sessions.

See also:

[meta_data_fetch_messages\(\)](#)

Parameters:

user a pointer to the meta user object requesting the messages update.

locked if set to META_NEED_LOCK, lock the specified meta user object for the duration of the request.

Returns:

-1 on failure, or 1 on success.

Definition at line 179 of file meta.c.

References `meta_data_fetch_messages()`, `meta_messages_update_sequences()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_MESSAGES`, `serial_get()`, and `serial_increment()`.

Referenced by `imap_session_update()`, `meta_get()`, and `sess_update()`.

5.207.3.15 void meta_messages_update_sequences (inx_t * *folders*, inx_t * *messages*)

Update the sequence numbers of a series of messages, each re-indexed by their containing folder.

Note:

All messages will be sequenced incrementally per folder, starting with a value of 1.

Parameters:

folders an inx holder containing all of the user's meta folder records.

messages an inx folder containing all the messages to be re-sequenced.

Returns:

This function returns no value.

Definition at line 83 of file meta.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), meta_folder_t, and meta_message_t.

Referenced by imap_append_message(), imap_close(), imap_expunge(), imap_message_copier(), meta_messages_copier(), meta_messages_login_update(), meta_messages_mover(), meta_messages_update(), meta_update_folders(), pop_session_destroy(), portal_endpoint_messages_copy(), portal_endpoint_messages_move(), and portal_endpoint_messages_remove().

5.207.4 Variable Documentation

5.207.4.1 message_header_t

Definition at line 113 of file messages.h.

Referenced by mail_load_message(), and mail_store_message_data().

5.207.4.2 message_t

Definition at line 100 of file messages.h.

Referenced by message_alloc(), and meta_data_fetch_folder_messages().

5.208 src/providers/prime/messages/messages.h File Reference

```
#include "chunks/chunks.h"
#include "parts/parts.h"
```

Functions

- [prime_message_t * encrypted_message_alloc](#) (void)
messages.c
- void [encrypted_message_cleanup](#) (prime_message_t *object)
- void [encrypted_message_free](#) (prime_message_t *object)
- [stringer_t * naked_message_get](#) (stringer_t *message, [prime_org_signet_t](#) *org, [prime_user_key_t](#) *user)
- [prime_message_t * naked_message_set](#) (stringer_t *message, [prime_org_key_t](#) *destination, [prime_user_signet_t](#) *recipient)

5.208.1 Function Documentation

5.208.1.1 [prime_message_t * encrypted_message_alloc](#) (void)

messages.c

Definition at line 57 of file messages.c.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_message_t](#).

Referenced by [naked_message_set\(\)](#).

5.208.1.2 void [encrypted_message_cleanup](#) (prime_message_t * *object*)

Definition at line 50 of file messages.c.

References [encrypted_message_free\(\)](#).

5.208.1.3 void [encrypted_message_free](#) (prime_message_t * *object*)

Definition at line 10 of file messages.c.

References [ed25519_free\(\)](#), [encrypted_chunk_free\(\)](#), [ephemeral_chunk_free\(\)](#), [log_pedantic](#), [mm_free\(\)](#), [secp256k1_free\(\)](#), and [st_free\(\)](#).

Referenced by [encrypted_message_cleanup\(\)](#), [naked_message_set\(\)](#), and [prime_free\(\)](#).

5.208.1.4 [stringer_t * naked_message_get](#) (stringer_t * *message*, [prime_org_signet_t](#) * *org*, [prime_user_key_t](#) * *user*)

Definition at line 73 of file messages.c.

References [chunk_header_read\(\)](#), [data](#), [encrypted_chunk_get\(\)](#), [ephemeral_chunk_cleanup\(\)](#), [ephemeral_chunk_free\(\)](#), [ephemeral_chunk_set\(\)](#), [keks_free\(\)](#), [keks_set\(\)](#), [mm_wipe\(\)](#), [part_decrypt\(\)](#), [pl_init\(\)](#), [pl_null\(\)](#), [PLACER](#), [placer_t](#), [PRIME_CHUNK_COMMON](#), [PRIME_CHUNK_EPHEMERAL](#), [PRIME_CHUNK_HEADERS](#), [prime_chunk_keks_t](#), [prime_chunk_keys_t](#), [prime_ephemeral_chunk_t](#), [prime_header_read\(\)](#), [PRIME_MESSAGE_NAKED](#), [PRIME_SIGNATURE_DESTINATION](#), [PRIME_SIGNATURE_TREE](#), [prime_signature_tree_t](#), [PRIME_SIGNATURE_USER](#), [signature_full_verify\(\)](#), [signature_tree_add\(\)](#), [signature_tree_alloc\(\)](#), [signature_tree_free\(\)](#), [signature_tree_verify\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [st_merge](#), and [type\(\)](#).

Referenced by [prime_message_decrypt\(\)](#).

5.208.1.5 `prime_message_t*` `naked_message_set` (`stringer_t *` *message*, `prime_org_key_t *` *destination*, `prime_user_signet_t *` *recipient*)

LOW: Effective, albeit kludgy, logic to ensure common headers are formatted correctly, and each header field is reformatted values reside on a single line.

Definition at line 228 of file `messages.c`.

References `common`, `ed25519_generate()`, `encrypted_chunk_buffer()`, `encrypted_chunk_set()`, `encrypted_message_alloc()`, `encrypted_message_free()`, `ephemeral_chunk_buffer()`, `ephemeral_chunk_get()`, `HEAP`, `JOINTED`, `keys_get()`, `length`, `mail_header_end()`, `mail_header_fetch_cleaned()`, `MANAGED_T`, `MANAGEDBUF`, `mm_copy()`, `part_buffer()`, `part_encrypt()`, `pl_init()`, `pl_null()`, `PLACER`, `placer_t`, `PRIME_CHUNK_BODY`, `PRIME_CHUNK_COMMON`, `PRIME_CHUNK_FLAG_NONE`, `PRIME_CHUNK_HEADERS`, `prime_encrypted_chunk_t`, `PRIME_MESSAGE_NAKED`, `prime_message_t`, `PRIME_SIGNATURE_DESTINATION`, `prime_signature_tree_t`, `PRIME_SIGNATURE_USER`, `secp256k1_generate()`, `secp256k1_public_get()`, `secp256k1_public_set()`, `signature_full_get()`, `signature_tree_add()`, `signature_tree_alloc()`, `signature_tree_free()`, `signature_tree_get()`, `st_alloc_opts()`, `st_append`, `st_cleanup`, `st_data_get()`, `st_free()`, `st_length_get()`, `st_merge`, `st_write`, and `type()`.

Referenced by `prime_message_encrypt()`.

5.209 src/objects/messages/meta.c File Reference

```
#include "magma.h"
```

Functions

- void [meta_message_free](#) ([meta_message_t](#) *message)
Free a meta message object (and its tags).
- [meta_message_t](#) * [meta_message_dupe](#) ([meta_message_t](#) *message)
Duplicate a meta message object (along with a deep copy of its tags).
- [meta_message_t](#) * [meta_message_by_number](#) ([inx_t](#) *messages, [uint64_t](#) number)
Retrieve a message by number.
- void [meta_messages_update_sequences](#) ([inx_t](#) *folders, [inx_t](#) *messages)
Update the sequence numbers of a series of messages, each re-indexed by their containing folder.
- [bool_t](#) [meta_messages_login_update](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Build a user's messages collection if it is empty, or needs to be refreshed (see note).
- [int_t](#) [meta_messages_update](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Refresh and resquence a user's message collection if it is stale.
- [bool_t](#) [meta_messages_copier](#) ([meta_user_t](#) *user, [meta_message_t](#) *message, [uint64_t](#) target, [uint64_t](#) *outnum, [bool_t](#) sequences, [META_LOCK_STATUS](#) locked)
- [int_t](#) [meta_messages_mover](#) ([meta_user_t](#) *user, [meta_message_t](#) *message, [uint64_t](#) target, [bool_t](#) lookup, [bool_t](#) sequences, [META_LOCK_STATUS](#) locked)

5.209.1 Function Documentation

5.209.1.1 [meta_message_t](#)* [meta_message_by_number](#) ([inx_t](#) * messages, [uint64_t](#) number)

Retrieve a message by number. meta.c

Parameters:

messages a pointer to the messages collection to be searched.

number the number of the message to be retrieved.

Returns:

NULL on failure, or a pointer to the meta message object of the matching mail message.

Definition at line 54 of file meta.c.

References [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), and [meta_message_t](#).

5.209.1.2 [meta_message_t](#)* [meta_message_dupe](#) ([meta_message_t](#) * message)

Duplicate a meta message object (along with a deep copy of its tags).

Parameters:

message the meta message object to be cloned.

Returns:

NULL on failure or a pointer to the duplicated meta message object on success.

Definition at line 33 of file meta.c.

References `ar_dupe()`, `meta_message_t`, and `mm_dupe()`.

Referenced by `imap_duplicate_messages()`, `imap_message_copier()`, `imap_search_messages()`, and `meta_messages_copier()`.

5.209.1.3 void meta_message_free (meta_message_t * message)

Free a meta message object (and its tags).

Returns:

This function returns no value.

Definition at line 14 of file meta.c.

References `ar_free()`, and `mm_free()`.

Referenced by `imap_duplicate_messages()`, `imap_search_messages()`, and `meta_data_fetch_messages()`.

5.209.1.4 bool_t meta_messages_copier (meta_user_t * user, meta_message_t * message, uint64_t target, uint64_t * outnum, bool_t sequences, META_LOCK_STATUS locked)**Parameters:**

Make a copy of a mail message, in the database and in memory.

See also:

[mail_copy_message\(\)](#)

Note:

The message copy will not have the deleted, hidden, or recent flags set in the database, but it will have the recent flag set in memory.

Parameters:

user a pointer to the meta user object to whom the message belong.

message the meta message object of the message to be copied.

target the numerical id of the folder to which the message will be copied.

outnum a pointer to an unsigned 64-bit integer that will receive the value of the numerical id of the message copy.

sequences if true, resequence the message folders after the copy has been made.

locked if set to `META_NEED_LOCK`, lock the specified meta user object for the duration of the request.

Returns:

true on success or false on failure.

Definition at line 238 of file meta.c.

References `inx_insert()`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `mail_copy_message()`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_HIDDEN`, `MAIL_STATUS_RECENT`, `meta_message_dupe()`, `meta_message_t`, `meta_messages_update_sequences()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `mm_free()`, `status`, `multi_t::u64`, and `multi_t::val`.

Referenced by `portal_endpoint_messages_copy()`.

5.209.1.5 **bool_t meta_messages_login_update** (meta_user_t * *user*, META_LOCK_STATUS *locked*)

Build a user's messages collection if it is empty, or needs to be refreshed (see note).

Note:

This function will fetch and sequence the user's messages from the database if the meta user object has no messages, or if the user has one or fewer pop sessions open and the messages serial number was stale.

Parameters:

user a pointer to the meta user object of the user requesting the messages update.

locked if set to META_NEED_LOCK, lock the specified meta user object for the duration of the request.

Returns:

true if no update was necessary or if the update was successful, or false otherwise.

Definition at line 125 of file meta.c.

References meta_data_fetch_messages(), meta_messages_update_sequences(), META_NEED_LOCK, meta_user_unlock(), meta_user_wlock(), OBJECT_MESSAGES, serial_get(), and serial_increment().

Referenced by pop_pass().

5.209.1.6 **int_t meta_messages_mover** (meta_user_t * *user*, meta_message_t * *message*, uint64_t *target*, bool_t *lookup*, bool_t *sequences*, META_LOCK_STATUS *locked*)

Parameters:

Move a mail message to another folder.

See also:

[mail_move_message\(\)](#)

Note:

The message will receive a copy of the recent flag in memory.

Parameters:

user a pointer to the meta user object to whom the message belong.

message the meta message object of the message to be moved.

target the numerical id of the folder to which the message will be copied.

lookup if set, verify the existence of the moved message in the user's messages.

sequences if true, resequence the message folders after the copy has been made.

locked if set to META_NEED_LOCK, lock the specified meta user object for the duration of the request.

Returns:

true on success or false on failure.

BUG: Currently we preserve the original message number which means when the target folder sequences are updated the moved message can appear anywhere in the list (since its based on message number). Inserting messages into the sequence list in this fashion will probably break clients. We may want to duplicate the row which would generate a new message number causing the newly added message to appear at the end of the folder listing. On the other hand POP ignores folders completely, so maybe it won't be quite so bad.

Definition at line 316 of file meta.c.

References inx_find(), log_pedantic, M_TYPE_UINT64, mail_move_message(), MAIL_STATUS_RECENT, meta_messages_update_sequences(), META_NEED_LOCK, meta_user_unlock(), meta_user_wlock(), multi_t::u64, and multi_t::val.

Referenced by portal_endpoint_messages_move().

5.209.1.7 `int_t meta_messages_update (meta_user_t * user, META_LOCK_STATUS locked)`

Refresh and resquence a user's message collection if it is stale.

Note:

The user's messages will only be updated if they are empty or if they are out of sync and the user has no open pop sessions.

See also:

[meta_data_fetch_messages\(\)](#)

Parameters:

user a pointer to the meta user object requesting the messages update.

locked if set to META_NEED_LOCK, lock the specified meta user object for the duration of the request.

Returns:

-1 on failure, or 1 on success.

Definition at line 179 of file meta.c.

References [meta_data_fetch_messages\(\)](#), [meta_messages_update_sequences\(\)](#), [META_NEED_LOCK](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_MESSAGES](#), [serial_get\(\)](#), and [serial_increment\(\)](#).

Referenced by [imap_session_update\(\)](#), [meta_get\(\)](#), and [sess_update\(\)](#).

5.209.1.8 `void meta_messages_update_sequences (inx_t * folders, inx_t * messages)`

Update the sequence numbers of a series of messages, each re-indexed by their containing folder.

Note:

All messages will be sequenced incrementally per folder, starting with a value of 1.

Parameters:

folders an inx holder containing all of the user's meta folder records.

messages an inx folder containing all the messages to be re-sequenced.

Returns:

This function returns no value.

Definition at line 83 of file meta.c.

References [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [meta_folder_t](#), and [meta_message_t](#).

Referenced by [imap_append_message\(\)](#), [imap_close\(\)](#), [imap_expunge\(\)](#), [imap_message_copier\(\)](#), [meta_messages_copier\(\)](#), [meta_messages_login_update\(\)](#), [meta_messages_mover\(\)](#), [meta_messages_update\(\)](#), [meta_update_folders\(\)](#), [pop_session_destroy\(\)](#), [portal_endpoint_messages_copy\(\)](#), [portal_endpoint_messages_move\(\)](#), and [portal_endpoint_messages_remove\(\)](#).

5.210 src/objects/meta/meta.c File Reference

```
#include "magma.h"
```

Functions

- void [meta_free](#) ([meta_user_t](#) *user)
Free a meta user object.
- [meta_user_t](#) * [meta_alloc](#) (void)
Allocate and initialize a meta user object.
- [int_t](#) [meta_get](#) (uint64_t usernum, [stringer_t](#) *username, [stringer_t](#) *salt, [stringer_t](#) *master, [stringer_t](#) *verification, [META_PROTOCOL](#) protocol, [META_GET](#) get, [meta_user_t](#) **output)
Lookup user and return their meta user object.

5.210.1 Function Documentation

5.210.1.1 [meta_user_t](#)* [meta_alloc](#) (void)

Allocate and initialize a meta user object. meta.c

Returns:

NULL on failure, or a pointer to the newly allocated meta user object on success.

Definition at line 47 of file meta.c.

References [log_pedantic](#), [meta_user_t](#), [mm_alloc\(\)](#), [mm_free\(\)](#), [mm_wipe\(\)](#), [mutex_init\(\)](#), [rwlock_attr_destroy\(\)](#), [rwlock_attr_init\(\)](#), [rwlock_attr_setkind\(\)](#), [rwlock_destroy\(\)](#), and [rwlock_init\(\)](#).

Referenced by [meta_inx_find\(\)](#).

5.210.1.2 void [meta_free](#) ([meta_user_t](#) * user)

Free a meta user object.

Parameters:

user a pointer to the meta user object to be destroyed.

Returns:

This function returns no value.

Definition at line 17 of file meta.c.

References [inx_cleanup\(\)](#), [mm_free\(\)](#), [mutex_destroy\(\)](#), [prime_cleanup\(\)](#), [rwlock_destroy\(\)](#), and [st_cleanup](#).

Referenced by [meta_inx_find\(\)](#), and [obj_cache_start\(\)](#).

5.210.1.3 [int_t](#) [meta_get](#) (uint64_t usernum, [stringer_t](#) * username, [stringer_t](#) * salt, [stringer_t](#) * master, [stringer_t](#) * verification, [META_PROTOCOL](#) protocol, [META_GET](#) get, [meta_user_t](#) ** output)

Lookup user and return their meta user object.

Note:

If the user is not found in the local session cache, the session will be constructed using the database, and then cached.

Parameters:

usernum the numeric identifier for the user account.

username the official username stored in the database.

salt the user specific salt value.

master the user account's master encryption key which will be used to unlock the private storage key.

verification the verification token.

protocol a set of protocol specifying the protocol used by the calling function. Values can be META_PROT_NONE, META_PROT_SMTP, META_PROT_POP, META_PROT_IMAP, META_PROT_WEB, or META_PROT_GENERIC.

get a set of protocol specifying the data to be retrieved (META_GET_NONE, META_GET_MESSAGES, META_GET_FOLDERS, or META_GET_CONTACTS)

output the address of a meta user object that will store a pointer to the result of the lookup.

Returns:

-1 on error, 0 on success, 1 for an authentication issue.

Definition at line 112 of file meta.c.

References log_pedantic, META_GET_ALIASES, META_GET_CONTACTS, META_GET_FOLDERS, META_GET_KEYS, META_GET_MESSAGES, meta_inx_find(), meta_inx_remove(), META_LOCKED, meta_messages_update(), meta_update_aliases(), meta_update_contacts(), meta_update_folders(), meta_update_keys(), meta_update_message_folders(), meta_update_realms(), meta_update_user(), meta_user_t, meta_user_unlock(), meta_user_wlock(), st_cleanup, st_cmp_cs_eq(), st_empty, and st_populated.

Referenced by api_endpoint_auth(), imap_login(), pop_pass(), and portal_endpoint_auth().

5.211 src/objects/meta/alerts.c File Reference

```
#include "magma.h"
```

Functions

- [meta_alert_t * alert_alloc](#) (uint64_t alertnum, [stringer_t](#) *type, [stringer_t](#) *message, uint64_t created)

Allocate and initialize a new user alert message object.

5.211.1 Function Documentation

5.211.1.1 meta_alert_t* alert_alloc (uint64_t alertnum, stringer_t * type, stringer_t * message, uint64_t created)

Allocate and initialize a new user alert message object. [alerts.c](#)

Parameters:

alertnum the numerical id of the user alert.

type a pointer to a managed string containing the type of the alert (e.g. "warning" or "alert").

message a pointer to a managed string containing the alert message.

created the UTC timecode for when the alert message was created.

Returns:

NULL on error or a pointer to the newly initialized user alert message object on success.

Definition at line 18 of file alerts.c.

References [align\(\)](#), [FOREIGNDATA](#), [JOINTED](#), [log_pedantic](#), [meta_alert_t](#), [mm_alloc\(\)](#), [mm_copy\(\)](#), [PLACER_T](#), [placer_t](#), [st_data_get\(\)](#), [st_length_get\(\)](#), and [STACK](#).

Referenced by [meta_data_fetch_alerts\(\)](#).

5.212 src/objects/meta/alias.c File Reference

```
#include "magma.h"
```

Functions

- `meta_alias_t * alias_alloc` (uint64_t *aliasnum*, `stringer_t` **address*, `stringer_t` **display*, `int_t` *selected*, uint64_t *created*)

Allocate and initialize a new user alias object.

5.212.1 Function Documentation

5.212.1.1 `meta_alias_t * alias_alloc` (uint64_t *aliasnum*, `stringer_t` **address*, `stringer_t` **display*, `int_t` *selected*, uint64_t *created*)

Allocate and initialize a new user alias object. [alias.c](#)

Parameters:

aliasnum the numerical identifier of the user alias.

address a managed string containing the email address associated with the alias.

display a managed string containing the display name associated with the alias.

selected a flag specifying whether this alias is the default selection for the user.

created a UNIX timestamp for the creation of this alias.

Returns:

NULL on failure, or a pointer to the newly initialized user alias object on success.

Definition at line 19 of file `alias.c`.

References `align()`, `FOREIGNDATA`, `JOINTED`, `log_pedantic`, `meta_alias_t`, `mm_alloc()`, `mm_copy()`, `PLACER_T`, `placer_t`, `st_data_get()`, `st_length_get()`, and `STACK`.

Referenced by `meta_data_fetch_mailbox_aliases()`.

5.213 src/objects/meta/crypto.c File Reference

```
#include "magma.h"
```

Functions

- [int_t meta_crypto_keys_create](#) (uint64_t usernum, [stringer_t](#) *username, [stringer_t](#) *realm, int64_t transaction)

Creates a new DIME key pair and insert it into the database.

5.213.1 Function Documentation

5.213.1.1 int_t meta_crypto_keys_create (uint64_t usernum, stringer_t * username, stringer_t * realm, int64_t transaction)

Creates a new DIME key pair and insert it into the database. crypto.c

Parameters:

user

master

Returns:

-1 if an error occurs, 0 if successful, and 1 if the insertion fails because a key already exists.

Definition at line 18 of file crypto.c.

References [BINARY](#), [log_info](#), [log_pedantic](#), [meta_data_insert_keys\(\)](#), [NONE](#), [org_key](#), [prime_cleanup\(\)](#), [prime_free\(\)](#), [prime_get\(\)](#), [prime_key_encrypt\(\)](#), [prime_key_generate\(\)](#), [prime_request_generate\(\)](#), [prime_request_sign\(\)](#), [prime_t](#), [PRIME_USER_KEY](#), [key_pair_t::private](#), [key_pair_t::public](#), [st_char_get\(\)](#), [st_cleanup](#), [st_free\(\)](#), and [st_length_int\(\)](#).

Referenced by [meta_update_keys\(\)](#), and [register_data_insert_user\(\)](#).

5.214 src/providers/dime/common/crypto.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <openssl/bio.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <openssl/rand.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include "core/core.h"
#include "providers/symbols.h"
#include "providers/prime/prime.h"
#include "dime/common/dcrypto.h"
#include "dime/common/misc.h"
#include "dime/common/error.h"
```

Functions

- `int _crypto_init (void)`
Initialize the cryptographic subsystem.
- `void _crypto_shutdown (void)`
Shutdown the cryptographic subsystem.
- `int _verify_ec_signature (unsigned char const *hash, size_t hlen, unsigned char const *sig, size_t slen, EC_KEY *key)`
Verify that an elliptic curve signature for a given hashed data buffer is valid.
- `int _verify_ec_sha_signature (unsigned char const *data, size_t dlen, unsigned int shabits, unsigned char const *sig, size_t slen, EC_KEY *key)`
Verify that a signature for a given data buffer is valid.
- `unsigned char * _ec_sign_data (unsigned char const *hash, size_t hlen, EC_KEY *key, size_t *siglen)`
Sign a body of data using the ECDSA algorithm.
- `unsigned char * _ec_sign_sha_data (unsigned char const *data, size_t dlen, unsigned int shabits, EC_KEY *key, size_t *siglen)`
Sign a SHA-hashed body of data using the ECDSA algorithm.
- `unsigned char * _serialize_ec_pubkey (EC_KEY *key, size_t *outsize)`
Serialize an EC public key to be shared.
- `EC_KEY * _deserialize_ec_pubkey (unsigned char const *buf, size_t blen)`
Deserialize an EC public key stored in binary format.
- `unsigned char * _serialize_ec_privkey (EC_KEY *key, size_t *outsize)`
Serialize an EC private key into a data buffer.
- `EC_KEY * _deserialize_ec_privkey (unsigned char const *buf, size_t blen)`

Deserialize an EC private key stored in binary format.

- `EC_KEY * _load_ec_privkey (char const *filename)`
Load an EC private key from a file.
- `EC_KEY * _load_ec_pubkey (char const *filename)`
Load an EC public key from a file.
- `EC_KEY * _generate_ec_keypair (void)`
Generate an EC key pair.
- `void _free_ec_key (EC_KEY *key)`
Free an EC keypair.
- `ED25519_KEY * _generate_ed25519_keypair (void)`
Generate an ed25519 key pair.
- `int _ed25519_sign_data (unsigned char const *data, size_t dlen, ED25519_KEY *key, ed25519_signature sigbuf)`
Take an ed25519 signature of a data buffer.
- `int _ed25519_verify_sig (unsigned char const *data, size_t dlen, ED25519_KEY *key, ed25519_signature sigbuf)`
Verify an ed25519 signature taken over a data buffer.
- `void _free_ed25519_key (ED25519_KEY *key)`
Free an ed25519 keypair.
- `void _free_ed25519_key_chain (ED25519_KEY **keys)`
Free a list of ed25519 keypairs.
- `ED25519_KEY * _load_ed25519_privkey (char const *filename)`
Load an ed25519 private key from a file.
- `ED25519_KEY * _deserialize_ed25519_pubkey (unsigned char const *serial_pubkey)`
Deserializes an ed25519 public key into a public-only ED25519_KEY structure that can only be used for signature verification, not signing.
- `ED25519_KEY * _deserialize_ed25519_privkey (unsigned char const *serial_privkey)`
Deserializes an ed25519 private key into a ED25519_KEY structure.
- `void * _ecies_env_derivation (void const *input, size_t ilen, void *output, size_t *olen)`
- `int _compute_aes256_kek (EC_KEY *public_key, EC_KEY *private_key, unsigned char *keybuf)`
Compute a derived AES-256 key from the intersection of a public EC key and a private EC key.
- `int _get_random_bytes (void *buf, size_t len)`
Fill a buffer with a sequence of (securely) random bytes.
- `int _encrypt_aes_256 (unsigned char *outbuf, unsigned char const *data, size_t dlen, unsigned char const *key, unsigned char const *iv)`
Encrypt a data buffer using an AES-256 key (in CBC mode).
- `int _decrypt_aes_256 (unsigned char *outbuf, unsigned char const *data, size_t dlen, unsigned char const *key, unsigned char const *iv)`
Decrypt a data buffer using an AES-256 key (in CBC mode).

5.214.1 Function Documentation

5.214.1.1 `int _compute_aes256_kek (EC_KEY * public_key, EC_KEY * private_key, unsigned char * keybuf)`

Compute a derived AES-256 key from the intersection of a public EC key and a private EC key.

Definition at line 720 of file `crypto.c`.

References `_ecies_env_derivation()`, `_secure_wipe()`, `EC_KEY_get0_public_key_d`, `ECDH_compute_key_d`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_INT`, and `SHA_512_SIZE`.

5.214.1.2 `int _crypto_init (void)`

Initialize the cryptographic subsystem.

Returns:

-1 if any part of the initialization process failed, or 0 on success.

Definition at line 28 of file `crypto.c`.

References `EC_ENCRYPT_CURVE`, `EC_GROUP_new_by_curve_name_d`, `ERR_UNSPEC`, `EVP_get_digestbyname_d`, `OBJ_nid2sn_d`, `OPENSSL_add_all_algorithms_noconf_d`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_INT`, `SSL_library_init_d`, and `SSL_load_error_strings_d`.

5.214.1.3 `void _crypto_shutdown (void)`

Shutdown the cryptographic subsystem.

Definition at line 52 of file `crypto.c`.

References `EC_GROUP_clear_free_d`, `ERR_free_strings_d`, and `EVP_cleanup_d`.

5.214.1.4 `int _decrypt_aes_256 (unsigned char * outbuf, unsigned char const * data, size_t dlen, unsigned char const * key, unsigned char const * iv)`

Decrypt a data buffer using an AES-256 key (in CBC mode).

Parameters:

outbuf a pointer to the output buffer that will receive the decrypted data. NOTE: the size of this buffer must be successfully negotiated by the caller.

data a pointer to the data buffer to be decrypted.

dlen the size, in bytes, of the data buffer to be decrypted.

key a pointer to the 32-byte buffer holding the AES-256 decryption key for the operation.

iv a pointer to the 32-byte initialization vector to be used for the decryption process.

Returns:

the number of bytes successfully decrypted on success, or -1 on failure.

Definition at line 849 of file `crypto.c`.

References `AES_256_PADDING_SIZE`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `EVP_aes_256_cbc_d`, `EVP_CIPHER_CTX_free_d`, `EVP_CIPHER_CTX_new_d`, `EVP_CIPHER_CTX_set_padding_d`, `EVP_DecryptFinal_ex_d`, `EVP_DecryptInit_ex_d`, `EVP_DecryptUpdate_d`, `PUSH_ERROR_OPENSSL`, and `RET_ERROR_INT`.

5.214.1.5 EC_KEY* _deserialize_ec_privkey (unsigned char const * *buf*, size_t *blen*)

Deserialize an EC private key stored in binary format.

Parameters:

buf a pointer to the buffer holding the EC private key in binary format.

blen the length, in bytes, of the buffer holding the EC private key.

Returns:

a pointer to the deserialized EC private key on success, or NULL on failure.

Definition at line 341 of file crypto.c.

References bufptr, ERR_BAD_PARAM, PLACER, RET_ERROR_PTR, and secp256k1_private_set().

5.214.1.6 EC_KEY* _deserialize_ec_pubkey (unsigned char const * *buf*, size_t *blen*)

Deserialize an EC public key stored in binary format.

Parameters:

buf a pointer to the buffer holding the EC public key in binary format.

blen the length, in bytes, of the buffer holding the EC public key.

Returns:

a pointer to the deserialized EC public key on success, or NULL on failure.

Definition at line 271 of file crypto.c.

References EC_GROUP_new_by_curve_name_d, EC_KEY_free_d, EC_KEY_new_d, EC_KEY_set_group_d, ERR_BAD_PARAM, ERR_UNSPEC, o2i_ECPublicKey_d, PUSH_ERROR_OPENSSL, and RET_ERROR_PTR.

Referenced by _load_ec_pubkey().

5.214.1.7 ED25519_KEY* _deserialize_ed25519_privkey (unsigned char const * *serial_privkey*)

Deserializes an ed25519 private key into a [ED25519_KEY](#) structure.

Parameters:

serial_privkey Serialized ed25519 private key.

Returns:

Pointer to the ED25119_KEY structure.

Definition at line 683 of file crypto.c.

References ED25519_KEY_SIZE, ed25519_publickey_donna(), ERR_BAD_PARAM, ERR_NOMEM, ED25519_KEY::private_key, ED25519_KEY::public_key, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

5.214.1.8 ED25519_KEY* _deserialize_ed25519_pubkey (unsigned char const * *serial_pubkey*)

Deserializes an ed25519 public key into a public-only [ED25519_KEY](#) structure that can only be used for signature verification, not signing.

Parameters:

serial_pubkey Serialized ed25519 public key.

Returns:

Pointer to [ED25519_KEY](#) structure.

Definition at line 657 of file crypto.c.

References [ED25519_KEY_SIZE](#), [ERR_BAD_PARAM](#), [ERR_NOMEM](#), [ED25519_KEY::public_key](#), [PUSH_ERROR_SYSCALL](#), and [RET_ERROR_PTR](#).

5.214.1.9 unsigned char* _ec_sign_data (unsigned char const * *hash*, size_t *hlen*, EC_KEY * *key*, size_t * *siglen*)

Sign a body of data using the ECDSA algorithm.

Parameters:

hash a pointer to the hashed data buffer to be signed.

hlen the length, in bytes, of the hashed data to be signed.

key the EC key which will have its private portion used to sign the supplied data.

siglen a pointer to a variable that will receive the length of the data signature buffer on success.

Returns:

NULL on failure, or a pointer to the newly allocated signature data buffer on success.

Definition at line 158 of file crypto.c.

References [ECDSA_do_sign_d](#), [ECDSA_SIG_free_d](#), [ERR_BAD_PARAM](#), [ERR_UNSPEC](#), [i2d_ECDSA_SIG_d](#), [PUSH_ERROR_OPENSSL](#), and [RET_ERROR_PTR](#).

Referenced by [_ec_sign_sha_data\(\)](#).

5.214.1.10 unsigned char* _ec_sign_sha_data (unsigned char const * *data*, size_t *dlen*, unsigned int *shabits*, EC_KEY * *key*, size_t * *siglen*)

Sign a SHA-hashed body of data using the ECDSA algorithm.

Parameters:

data a pointer to the data buffer to be signed.

dlen the length, in bytes, of the data buffer to be signed.

shabits the number of bits for the desired SHA hash (160, 256, or 512).

key the EC key which will have its private portion used to sign the supplied data.

siglen a pointer to a variable that will receive the length of the data signature buffer on success.

Returns:

NULL on failure, or a pointer to the newly allocated signature data buffer on success.

Definition at line 204 of file crypto.c.

References [_compute_sha_hash\(\)](#), [_ec_sign_data\(\)](#), [ERR_BAD_PARAM](#), [ERR_UNSPEC](#), [RET_ERROR_PTR](#), and [SHA_512_SIZE](#).

5.214.1.11 void* **_ecies_env_derivation** (void const * *input*, size_t *ilen*, void * *output*, size_t * *olen*)

Note:

This function was taken from [providers/cryptography/openssl.c](#)

Definition at line 706 of file crypto.c.

References EVP_Digest_d.

Referenced by _compute_aes256_kek().

5.214.1.12 int **_ed25519_sign_data** (unsigned char const * *data*, size_t *dlen*, ED25519_KEY * *key*, ed25519_signature *sigbuf*)

Take an ed25519 signature of a data buffer.

Returns:

0 on success or -1 on failure.

Definition at line 529 of file crypto.c.

References ed25519_sign_donna(), ERR_BAD_PARAM, ED25519_KEY::private_key, ED25519_KEY::public_key, and RET_ERROR_INT.

5.214.1.13 int **_ed25519_verify_sig** (unsigned char const * *data*, size_t *dlen*, ED25519_KEY * *key*, ed25519_signature *sigbuf*)

Verify an ed25519 signature taken over a data buffer.

Returns:

1 if the signature matched the buffer, 0 if it did not, or -1 on failure.

Definition at line 545 of file crypto.c.

References ed25519_sign_open_donna(), ERR_BAD_PARAM, ED25519_KEY::public_key, and RET_ERROR_INT.

Referenced by _verify_dx_certificate().

5.214.1.14 int **_encrypt_aes_256** (unsigned char * *outbuf*, unsigned char const * *data*, size_t *dlen*, unsigned char const * *key*, unsigned char const * *iv*)

Encrypt a data buffer using an AES-256 key (in CBC mode).

Parameters:

outbuf a pointer to the output buffer that will receive the encrypted data. NOTE: the size of this buffer must be successfully negotiated by the caller.

data a pointer to the data buffer to be encrypted.

dlen the size, in bytes, of the data buffer to be encrypted.

key a pointer to the 32-byte buffer holding the AES-256 encryption key for the operation.

iv a pointer to the 32-byte initialization vector to be used for the encryption process.

Returns:

the number of bytes successfully encrypted on success, or -1 on failure.

Definition at line 784 of file crypto.c.

References AES_256_PADDING_SIZE, ERR_BAD_PARAM, ERR_UNSPEC, EVP_aes_256_cbc_d, EVP_CIPHER_CTX_free_d, EVP_CIPHER_CTX_new_d, EVP_CIPHER_CTX_set_padding_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, PUSH_ERROR_OPENSSL, and RET_ERROR_INT.

5.214.1.15 void _free_ec_key (EC_KEY * *key*)

Free an EC keypair.

Parameters:

key a pointer to the EC keypair to be freed.

Definition at line 484 of file crypto.c.

References EC_KEY_free_d.

5.214.1.16 void _free_ed25519_key (ED25519_KEY * *key*)

Free an ed25519 keypair.

Parameters:

key a pointer to the ed25519 keypair to be freed.

Definition at line 567 of file crypto.c.

References _secure_wipe().

Referenced by _free_ed25519_key_chain().

5.214.1.17 void _free_ed25519_key_chain (ED25519_KEY ** *keys*)

Free a list of ed25519 keypairs.

Parameters:

keys a pointer to a NULL terminated list of ed25519 keypair to be freed.

Definition at line 582 of file crypto.c.

References _free_ed25519_key().

5.214.1.18 EC_KEY* _generate_ec_keypair (void)

Generate an EC key pair.

Returns:

a newly allocated and generated EC key pair on success, or NULL on failure.

Definition at line 446 of file crypto.c.

References secp256k1_generate().

5.214.1.19 ED25519_KEY* _generate_ed25519_keypair (void)

Generate an ed25519 key pair.

Returns:

a newly allocated and generated ed25519 key pair on success, or NULL on failure.

Definition at line 500 of file crypto.c.

References _secure_wipe(), ed25519_publickey_donna(), ERR_NOMEM, ERR_UNSPEC, ED25519_KEY::private_key, ED25519_KEY::public_key, PUSH_ERROR_OPENSSL, PUSH_ERROR_SYSCALL, RAND_bytes_d, and RET_ERROR_PTR.

5.214.1.20 `int _get_random_bytes (void * buf, size_t len)`

Fill a buffer with a sequence of (securely) random bytes.

Parameters:

buf a pointer to the buffer to be filled with random bytes.

len the length, in bytes, of the buffer to be filled.

Returns:

0 on success or -1 on failure.

Definition at line 752 of file crypto.c.

References ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_OPENSSL, RAND_bytes_d, and RET_ERROR_INT.

5.214.1.21 `EC_KEY* _load_ec_privkey (char const * filename)`

Load an EC private key from a file.

Parameters:

filename the name of the filename from which the key should be loaded

Returns:

a pointer to the deserialized private key from the the file.

Definition at line 369 of file crypto.c.

References _b64decode(), _read_pem_data(), _secure_wipe(), ERR_BAD_PARAM, ERR_UNSPEC, PLACER, RET_ERROR_PTR, and secp256k1_private_set().

5.214.1.22 `EC_KEY* _load_ec_pubkey (char const * filename)`

Load an EC public key from a file.

Parameters:

filename the name of the file from which the key should be loaded

Returns:

a pointer to the deserialized public key from the the file.

Definition at line 409 of file crypto.c.

References _b64decode(), _deserialize_ec_pubkey(), _read_pem_data(), _secure_wipe(), ERR_BAD_PARAM, ERR_UNSPEC, and RET_ERROR_PTR.

5.214.1.23 `ED25519_KEY* _load_ed25519_privkey (char const * filename)`

Load an ed25519 private key from a file.

Parameters:

filename the path of the armored file from which the ed25519 private key will be loaded.

Returns:

a pointer to a newly allocated ed25519 keypair on success, or NULL on failure.

Definition at line 604 of file crypto.c.

References `_b64decode()`, `_read_pem_data()`, `_secure_wipe()`, `ED25519_KEY_SIZE`, `ed25519_publickey_donna()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `ED25519_KEY::private_key`, `ED25519_KEY::public_key`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

5.214.1.24 unsigned char* _serialize_ec_privkey (EC_KEY *key, size_t *outsize)

Serialize an EC private key into a data buffer.

Parameters:

key a pointer to the EC key pair to have its private key serialized.

outsize a pointer to a variable that will receive the length of the serialized key on success.

Returns:

a pointer to the serialized EC private key on success, or NULL on failure.

Definition at line 309 of file crypto.c.

References `ERR_BAD_PARAM`, `ERR_UNSPEC`, `mm_alloc()`, `PLACER`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, and `secp256k1_private_get()`.

5.214.1.25 unsigned char* _serialize_ec_pubkey (EC_KEY *key, size_t *outsize)

Serialize an EC public key to be shared.

Parameters:

key a pointer to the EC key pair to have its public key serialized.

outsize a pointer to a variable that will receive the length of the serialized key on success.

Returns:

a pointer to the serialized EC public key on success, or NULL on failure.

Definition at line 232 of file crypto.c.

References `ERR_BAD_PARAM`, `ERR_UNSPEC`, `MANAGEDBUF`, `mm_alloc()`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, `secp256k1_public_get()`, `st_data_get()`, and `st_length_get()`.

5.214.1.26 int _verify_ec_sha_signature (unsigned char const *data, size_t dlen, unsigned int shabits, unsigned char const *sig, size_t slen, EC_KEY *key)

Verify that a signature for a given data buffer is valid.

Parameters:

data a pointer to the data buffer used to generate the signature.

dlen the length, in bytes, of the data buffer.

shabits the number of bits for the desired SHA hash (160, 256, or 512).

sig a pointer to the signature buffer to be verified against the input data.

slen the length, in bytes, of the signature buffer.

key the EC key which will have its public portion used to verify the signature of the supplied data.

Returns:

-1 on general failure, 0 if the signature did not match the data, or 1 if it did.

Definition at line 125 of file crypto.c.

References `_compute_sha_hash()`, `_verify_ec_signature()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `RET_ERROR_INT`, and `SHA_512_SIZE`.

5.214.1.27 int _verify_ec_signature (unsigned char const * hash, size_t hlen, unsigned char const * sig, size_t slen, EC_KEY * key)

Verify that an elliptic curve signature for a given hashed data buffer is valid.

Parameters:

hash a pointer to the hashed data buffer used to generate the signature.

hlen the length, in bytes, of the hashed data buffer.

sig a pointer to the signature buffer to be verified against the input data.

slen the length, in bytes, of the signature buffer.

key the EC key which will have its public portion used to verify the signature of the supplied hashed data.

Returns:

-1 on general failure, 0 if the signature did not match the hashed data, or 1 if it did.

Definition at line 81 of file crypto.c.

References `d2i_ECDSA_SIG_d`, `ECDSA_do_verify_d`, `ECDSA_SIG_free_d`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, and `RET_ERROR_INT`.

Referenced by `_verify_ec_sha_signature()`.

5.215 src/providers/stacie/crypto.c File Reference

Use a realm key to encrypt/decrypt data. `#include "magma.h"`

Functions

- [stringer_t * stacie_encrypt](#) (uint16_t serial, [stringer_t *vector_key](#), [stringer_t *tag_key](#), [stringer_t *cipher_key](#), [stringer_t *buffer](#))
- [stringer_t * stacie_decrypt](#) ([stringer_t *vector_key](#), [stringer_t *tag_key](#), [stringer_t *cipher_key](#), [stringer_t *buffer](#))

crypto.c

5.215.1 Detailed Description

Use a realm key to encrypt/decrypt data.

Definition in file [crypto.c](#).

5.215.2 Function Documentation

5.215.2.1 [stringer_t* stacie_decrypt](#) ([stringer_t * vector_key](#), [stringer_t * tag_key](#), [stringer_t * cipher_key](#), [stringer_t * buffer](#))

crypto.c

Parameters:

vector_key

tag_key

cipher_key

buffer

Returns:

Definition at line 185 of file *crypto.c*.

References [EVP_aes_256_gcm_d](#), [EVP_CIPHER_CTX_cleanup_d](#), [EVP_CIPHER_CTX_ctrl_d](#), [EVP_CIPHER_CTX_init_d](#), [EVP_CIPHER_CTX_key_length_d](#), [EVP_DecryptFinal_ex_d](#), [EVP_DecryptInit_ex_d](#), [EVP_DecryptUpdate_d](#), [log_error](#), [log_pedantic](#), [MANAGEDBUF](#), [MEMORYBUF](#), [mm_move\(\)](#), [PLACER](#), [ssl_error_string\(\)](#), [st_alloc\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [st_length_set\(\)](#), [st_xor\(\)](#), and [STACIE_ENVELOPE_LENGTH](#).

5.215.2.2 [stringer_t* stacie_encrypt](#) (uint16_t *serial*, [stringer_t * vector_key](#), [stringer_t * tag_key](#), [stringer_t * cipher_key](#), [stringer_t * buffer](#))

Parameters:

serial

vector_key

tag_key

cipher_key

buffer

Returns:

Definition at line 21 of file crypto.c.

References `EVP_aes_256_gcm_d`, `EVP_CIPHER_CTX_cleanup_d`, `EVP_CIPHER_CTX_ctrl_d`, `EVP_CIPHER_CTX_init_d`, `EVP_CIPHER_CTX_key_length_d`, `EVP_EncryptFinal_ex_d`, `EVP_EncryptInit_ex_d`, `EVP_EncryptUpdate_d`, `log_error`, `log_pedantic`, `MANAGEDBUF`, `mm_move()`, `PLACER`, `rand_write()`, `st_alloc()`, `st_data_get()`, `st_empty_out()`, `st_free()`, `st_length_get()`, `st_length_set()`, `st_xor()`, `STACIE_BLOCK_LENGTH`, `STACIE_ENCRYPT_MAX`, and `STACIE_ENVELOPE_LENGTH`.

5.216 src/objects/meta/indexes.c File Reference

```
#include "magma.h"
```

Functions

- void [meta_inx_remove](#) (uint64_t usernum, [META_PROTOCOL](#) protocol)
Lock a user's object in the cache and decrement their reference counter.
- [meta_user_t](#) * [meta_inx_find](#) (uint64_t usernum, [META_PROTOCOL](#) protocol)
[indexes.c](#)

5.216.1 Function Documentation

5.216.1.1 [meta_user_t](#)* [meta_inx_find](#) (uint64_t *usernum*, [META_PROTOCOL](#) *protocol*)

[indexes.c](#)

Definition at line 40 of file [indexes.c](#).

References [inx_find\(\)](#), [inx_insert\(\)](#), [inx_lock_write\(\)](#), [inx_unlock\(\)](#), [M_TYPE_UINT64](#), [object_cache_t::meta](#), [meta_alloc\(\)](#), [meta_free\(\)](#), [meta_user_ref_add\(\)](#), [meta_user_t](#), and [objects](#).

Referenced by [meta_get\(\)](#).

5.216.1.2 void [meta_inx_remove](#) (uint64_t *usernum*, [META_PROTOCOL](#) *protocol*)

Lock a user's object in the cache and decrement their reference counter.

See also:

[meta_user_ref_dec\(\)](#)

Parameters:

username a managed string containing the name of the user to be adjusted.

protocol specifies the protocol bound to the reference counter to be decremented ([META_PROT_WEB](#), [META_PROT_IMAP](#), etc.)

Returns:

This function returns no value.

Definition at line 17 of file [indexes.c](#).

References [inx_find\(\)](#), [inx_lock_read\(\)](#), [inx_unlock\(\)](#), [M_TYPE_UINT64](#), [object_cache_t::meta](#), [meta_user_ref_dec\(\)](#), [meta_user_t](#), and [objects](#).

Referenced by [imap_login\(\)](#), [imap_session_destroy\(\)](#), [meta_get\(\)](#), [pop_pass\(\)](#), [pop_session_destroy\(\)](#), [portal_endpoint_auth\(\)](#), and [sess_destroy\(\)](#).

5.217 src/objects/meta/locking.c File Reference

```
#include "magma.h"
```

Functions

- void [meta_user_rlock](#) ([meta_user_t](#) *user)
Acquire a read lock for a meta user object.
- void [meta_user_wlock](#) ([meta_user_t](#) *user)
Acquire a write lock for a meta user object.
- void [meta_user_unlock](#) ([meta_user_t](#) *user)
Release the lock for a meta user object.

5.217.1 Function Documentation

5.217.1.1 void [meta_user_rlock](#) ([meta_user_t](#) * user)

Acquire a read lock for a meta user object. [locking.c](#)

Parameters:

user a pointer to the meta user object to be locked.

Returns:

This function returns no value.

Definition at line 15 of file [locking.c](#).

References [rwlock_lock_read\(\)](#).

Referenced by [imap_close\(\)](#), [imap_examine\(\)](#), [imap_expunge\(\)](#), [imap_fetch\(\)](#), [imap_list\(\)](#), [imap_login\(\)](#), [imap_lsub\(\)](#), [imap_search_messages\(\)](#), [imap_status\(\)](#), [pop_last\(\)](#), [pop_list\(\)](#), [pop_retr\(\)](#), [pop_stat\(\)](#), [pop_top\(\)](#), [pop_uidl\(\)](#), [portal_endpoint_contacts_add\(\)](#), [portal_endpoint_contacts_copy\(\)](#), [portal_endpoint_contacts_edit\(\)](#), [portal_endpoint_folders_tags\(\)](#), [portal_endpoint_messages_flag\(\)](#), [portal_endpoint_messages_list\(\)](#), and [portal_endpoint_messages_tag\(\)](#).

5.217.1.2 void [meta_user_unlock](#) ([meta_user_t](#) * user)

Release the lock for a meta user object.

Parameters:

user a pointer to the meta user object to be unlocked.

Returns:

This function returns no value.

Definition at line 47 of file [locking.c](#).

References [rwlock_unlock\(\)](#).

Referenced by `imap_append()`, `imap_close()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_list()`, `imap_login()`, `imap_lsub()`, `imap_rename()`, `imap_search_messages()`, `imap_select()`, `imap_session_destroy()`, `imap_session_update()`, `imap_status()`, `imap_store()`, `imap_subscribe()`, `meta_get()`, `meta_messages_copier()`, `meta_messages_login_update()`, `meta_messages_mover()`, `meta_messages_update()`, `meta_update_aliases()`, `meta_update_contacts()`, `meta_update_folders()`, `meta_update_keys()`, `meta_update_message_folders()`, `meta_update_realms()`, `meta_update_user()`, `pop_dele()`, `pop_last()`, `pop_list()`, `pop_pass()`, `pop_retr()`, `pop_session_destroy()`, `pop_session_reset()`, `pop_stat()`, `pop_top()`, `pop_uidl()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_tag()`, `portal_folder_contacts_add()`, `portal_folder_contacts_remove()`, `portal_folder_mail_add()`, and `portal_folder_mail_remove()`.

5.217.1.3 void meta_user_wlock (meta_user_t * user)

Acquire a write lock for a meta user object.

Parameters:

user a pointer to the meta user object to be locked.

Returns:

This function returns no value.

Definition at line 31 of file `locking.c`.

References `rwlock_lock_write()`.

Referenced by `imap_append()`, `imap_close()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_expunge()`, `imap_fetch()`, `imap_rename()`, `imap_select()`, `imap_session_destroy()`, `imap_session_update()`, `imap_store()`, `imap_subscribe()`, `meta_get()`, `meta_messages_copier()`, `meta_messages_login_update()`, `meta_messages_mover()`, `meta_messages_update()`, `meta_update_aliases()`, `meta_update_contacts()`, `meta_update_folders()`, `meta_update_keys()`, `meta_update_message_folders()`, `meta_update_realms()`, `meta_update_user()`, `pop_dele()`, `pop_pass()`, `pop_session_destroy()`, `pop_session_reset()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_folders_rename()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_tag()`, `portal_folder_contacts_add()`, `portal_folder_contacts_remove()`, `portal_folder_mail_add()`, and `portal_folder_mail_remove()`.

5.218 src/objects/meta/references.c File Reference

```
#include "magma.h"
```

Functions

- void `meta_user_ref_add` (`meta_user_t` *user, `META_PROTOCOL` protocol)
Increment a meta user's reference counter for a specified protocol and update the activity timestamp.
- void `meta_user_ref_dec` (`meta_user_t` *user, `META_PROTOCOL` protocol)
Decrement a user's reference counter for the specified protocol and update the activity timestamp.
- uint64_t `meta_user_ref_protocol_total` (`meta_user_t` *user, `META_PROTOCOL` protocol)
Get the total reference count for the user object over the specified protocol.
- uint64_t `meta_user_ref_total` (`meta_user_t` *user)
Get the total reference count for a user object, over all of the supported protocols.
- time_t `meta_user_ref_stamp` (`meta_user_t` *user)
Get the activity timestamp for a meta user object.

5.218.1 Function Documentation

5.218.1.1 void meta_user_ref_add (meta_user_t * user, META_PROTOCOL protocol)

Increment a meta user's reference counter for a specified protocol and update the activity timestamp. [references.c](#)

Note:

`META_PROTOCOL_GENERIC` can be specified for non-specific, protocol independent accounting purposes.

Parameters:

user a pointer to the meta user object to be adjusted.
protocol the protocol identifier for the session.

Returns:

This function returns no value.

Definition at line 20 of file references.c.

References `log_pedantic`, `META_PROTOCOL_GENERIC`, `META_PROTOCOL_IMAP`, `META_PROTOCOL_POP`, `META_PROTOCOL_SMTP`, `META_PROTOCOL_WEB`, `mutex_lock()`, and `mutex_unlock()`.

Referenced by `meta_inx_find()`.

5.218.1.2 void meta_user_ref_dec (meta_user_t * user, META_PROTOCOL protocol)

Decrement a user's reference counter for the specified protocol and update the activity timestamp.

Note:

`META_PROTOCOL_GENERIC` can be specified for non-specific, protocol independent accounting purposes.

Parameters:

user a pointer to the meta user object to be adjusted.

protocol the protocol identifier for the session using the META_PROTOCOL enumerator.

Returns:

This function returns no value.

Definition at line 60 of file references.c.

References log_pedantic, META_PROTOCOL_GENERIC, META_PROTOCOL_IMAP, META_PROTOCOL_POP, META_PROTOCOL_SMTP, META_PROTOCOL_WEB, mutex_lock(), and mutex_unlock().

Referenced by meta_inx_remove(), and portal_endpoint_logout().

5.218.1.3 uint64_t meta_user_ref_protocol_total (meta_user_t * user, META_PROTOCOL protocol)

Get the total reference count for the user object over the specified protocol.

Parameters:

user a pointer to the meta user object to be examined.

protocol the protocol identifier for the session using the META_PROTOCOL enumerator.

Returns:

the total number of references held by the user.

Definition at line 98 of file references.c.

References log_pedantic, META_PROTOCOL_GENERIC, META_PROTOCOL_IMAP, META_PROTOCOL_POP, META_PROTOCOL_SMTP, META_PROTOCOL_WEB, mutex_lock(), and mutex_unlock().

Referenced by pop_pass().

5.218.1.4 time_t meta_user_ref_stamp (meta_user_t * user)

Get the activity timestamp for a meta user object.

Parameters:

user a pointer to the meta user object to be examined.

Returns:

a timestamp containing the last time the meta user object's reference count changed.

Definition at line 161 of file references.c.

References mutex_lock(), and mutex_unlock().

Referenced by obj_cache_prune().

5.218.1.5 uint64_t meta_user_ref_total (meta_user_t * user)

Get the total reference count for a user object, over all of the supported protocols.

Parameters:

user a pointer to the meta user object to be examined.

Returns:

the total number of references held by the user.

Definition at line 134 of file references.c.

References mutex_lock(), and mutex_unlock().

Referenced by obj_cache_prune().

5.219 src/objects/meta/serials.c File Reference

```
#include "magma.h"
```

Functions

- void [meta_user_serial_set](#) ([meta_user_t](#) *user, uint64_t object, uint64_t serial)
Set an object serial number for a given meta user structure.
- uint64_t [meta_user_serial_get](#) ([meta_user_t](#) *user, uint64_t object)
Get an object serial number for a given meta object.
- bool_t [meta_user_serial_check](#) ([meta_user_t](#) *user, uint64_t object)
Check an object's serial number to see if it is up-to-date.

5.219.1 Function Documentation

5.219.1.1 bool_t meta_user_serial_check (meta_user_t * user, uint64_t object)

Check an object's serial number to see if it is up-to-date. serials.c

Note:

The object's serial number will be incremented regardless of whether it is consistent with the cache.

Parameters:

user the meta user object to whom the object belongs.
object the serial object to be checked for changes.

Returns:

0 if the object did not to be refreshed, or 1 if it doesn't match the internal checkpoint and should be updated.

Definition at line 86 of file serials.c.

References [meta_user_serial_get\(\)](#), [meta_user_serial_set\(\)](#), [serial_get\(\)](#), and [serial_increment\(\)](#).

Referenced by [portal_endpoint_messages_tag\(\)](#), and [sess_serial_check\(\)](#).

5.219.1.2 uint64_t meta_user_serial_get (meta_user_t * user, uint64_t object)

Get an object serial number for a given meta object.

Parameters:

user the meta user structure to be examined.
object the object to query for the serial number.

Returns:

the serial number value of the specified object, or 0 on failure. LOW: Swap the object and user params so the interface matches the [serial_get/set](#) functions.

Definition at line 53 of file serials.c.

References [OBJECT_ALIASES](#), [OBJECT_CONTACTS](#), [OBJECT_FOLDERS](#), [OBJECT_MESSAGES](#), and [OBJECT_USER](#).

Referenced by [meta_update_aliases\(\)](#), [meta_update_user\(\)](#), and [meta_user_serial_check\(\)](#).

5.219.1.3 void meta_user_serial_set (meta_user_t * *user*, uint64_t *object*, uint64_t *serial*)

Set an object serial number for a given meta user structure.

Parameters:

user the meta user object to be adjusted.

object the object to receive the new serial number.

serial the new serial number to be set.

Returns:

This function returns no value.

Definition at line 20 of file serials.c.

References OBJECT_ALIASES, OBJECT_CONTACTS, OBJECT_FOLDERS, OBJECT_MESSAGES, and OBJECT_USER.

Referenced by meta_update_aliases(), meta_update_user(), and meta_user_serial_check().

5.220 src/objects/serials.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * serial_prefix](#) (uint64_t type)
Return a textual prefix describing an object type.
- uint64_t [serial_get](#) (uint64_t type, uint64_t num)
Get the serial number (checkpoint value) for an object from memcached.
- uint64_t [serial_increment](#) (uint64_t type, uint64_t num)
Increment the serial number for an object in memcached.
- uint64_t [serial_reset](#) (uint64_t type, uint64_t num)
Reset the serial number to 1 for an object in memcached.

Variables

- [stringer_t * serial_prefix_strings](#) []

5.220.1 Function Documentation

5.220.1.1 uint64_t serial_get (uint64_t type, uint64_t num)

Get the serial number (checkpoint value) for an object from memcached. serials.c

Parameters:

- type** the serial type to be queried (OBJECT_USER, OBJECT_CONFIG, OBJECT_FOLDERS, OBJECT_MESSAGES, or OBJECT_CONTACTS).
- num** the specific object identifier.

Returns:

- 0 on failure or the serial number of the requested object.

Definition at line 62 of file serials.c.

References [cache_increment\(\)](#), [log_pedantic](#), [serial_prefix\(\)](#), [st_aprint\(\)](#), [st_char_get\(\)](#), [st_free\(\)](#), and [st_length_int\(\)](#).

Referenced by [imap_append_message\(\)](#), [imap_close\(\)](#), [imap_create\(\)](#), [imap_delete\(\)](#), [imap_expunge\(\)](#), [imap_fetch\(\)](#), [imap_message_copier\(\)](#), [imap_rename\(\)](#), [imap_session_update\(\)](#), [imap_store\(\)](#), [meta_messages_login_update\(\)](#), [meta_messages_update\(\)](#), [meta_update_aliases\(\)](#), [meta_update_contacts\(\)](#), [meta_update_folders\(\)](#), [meta_update_message_folders\(\)](#), [meta_update_user\(\)](#), [meta_user_serial_check\(\)](#), [portal_endpoint_folders_rename\(\)](#), [portal_endpoint_messages_copy\(\)](#), [portal_endpoint_messages_flag\(\)](#), [portal_endpoint_messages_move\(\)](#), [portal_endpoint_messages_remove\(\)](#), [user_config_create\(\)](#), and [user_config_update\(\)](#).

5.220.1.2 uint64_t serial_increment (uint64_t type, uint64_t num)

Increment the serial number for an object in memcached.

Parameters:

- type* the serial type to be queried (OBJECT_USER, OBJECT_CONFIG, OBJECT_FOLDERS, OBJECT_MESSAGES, or OBJECT_CONTACTS).
- num* the specific object identifier.

Returns:

- 0 on failure or the new serial number of the requested object.

Definition at line 86 of file serials.c.

References cache_increment(), log_pedantic, serial_prefix(), st_aprint(), st_char_get(), st_free(), and st_length_int().

Referenced by contact_move(), imap_append_message(), imap_close(), imap_create(), imap_delete(), imap_expunge(), imap_fetch(), imap_message_copier(), imap_rename(), imap_store(), mail_load_message(), meta_messages_login_update(), meta_messages_update(), meta_update_aliases(), meta_update_contacts(), meta_update_folders(), meta_update_message_folders(), meta_update_user(), meta_user_serial_check(), pop_session_destroy(), portal_endpoint_folders_rename(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), smtp_rollout(), smtp_store_message(), and user_config_edit().

5.220.1.3 stringer_t* serial_prefix (uint64_t type)

Return a textual prefix describing an object type.

Parameters:

- type* the serial object type: OBJECT_USER, OBJECT_CONFIG, OBJECT_FOLDERS, OBJECT_MESSAGES, or OBJECT_CONTACTS.

Returns:

- a managed string containing a description of the object type, or NULL on failure.

Definition at line 24 of file serials.c.

References log_pedantic, OBJECT_ALIASES, OBJECT_CONFIG, OBJECT_CONTACTS, OBJECT_FOLDERS, OBJECT_MESSAGES, OBJECT_USER, and serial_prefix_strings.

Referenced by serial_get(), serial_increment(), and serial_reset().

5.220.1.4 uint64_t serial_reset (uint64_t type, uint64_t num)

Reset the serial number to 1 for an object in memcached.

Parameters:

- type* the serial type to be queried (OBJECT_USER, OBJECT_CONFIG, OBJECT_FOLDERS, OBJECT_MESSAGES, or OBJECT_CONTACTS).
- num* the specific object identifier.

Returns:

- 0 on failure or the new value of the cached serial number (1) on success.

Definition at line 110 of file serials.c.

References cache_set(), CONSTANT, log_pedantic, serial_prefix(), st_aprint(), st_char_get(), st_free(), and st_length_int().

5.220.2 Variable Documentation

5.220.2.1 stringer_t* serial_prefix_strings[]

Initial value:

```
{  
    CONSTANT("user"),  
    CONSTANT("config"),  
    CONSTANT("folders"),  
    CONSTANT("messages"),  
    CONSTANT("contacts"),  
}
```

Definition at line 10 of file serials.c.

Referenced by serial_prefix().

5.221 src/objects/meta/updaters.c File Reference

```
#include "magma.h"
```

Functions

- [int_t meta_update_realms](#) ([meta_user_t](#) *user, [stringer_t](#) *salt, [stringer_t](#) *master, [META_LOCK_STATUS](#) locked)
Fetches the user realm keys and extracts the different components.
- [int_t meta_update_keys](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Fetches the user signet and private key. Relies on the mail realm to decrypt the values.
- [int_t meta_update_user](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Build a user's meta information from specified data parameters.
- [int_t meta_update_aliases](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Pulls the list of user mailboxes and their display names from the database.
- [int_t meta_update_contacts](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Update a user's contacts if necessary.
- [int_t meta_update_folders](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Update a user's message folders if necessary.
- [int_t meta_update_message_folders](#) ([meta_user_t](#) *user, [META_LOCK_STATUS](#) locked)
Refresh a user's message folders if stale, or retrieve them from the database if they are not in memory.

5.221.1 Function Documentation

5.221.1.1 [int_t meta_update_aliases](#) ([meta_user_t](#) * user, [META_LOCK_STATUS](#) locked)

Pulls the list of user mailboxes and their display names from the database. [updaters.c](#)

Parameters:

user a pointer to the meta object that is to be populated.

locked the meta lock status of the operation (if [META_NEED_LOCK](#) is supplied, the meta user object will be locked for the duration of the function.

Returns:

Definition at line 261 of file [updaters.c](#).

References [meta_data_fetch_mailbox_aliases\(\)](#), [META_NEED_LOCK](#), [meta_user_serial_get\(\)](#), [meta_user_serial_set\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_ALIASES](#), [serial_get\(\)](#), and [serial_increment\(\)](#).

Referenced by [meta_get\(\)](#), and [sess_update\(\)](#).

5.221.1.2 `int_t meta_update_contacts (meta_user_t * user, META_LOCK_STATUS locked)`

Update a user's contacts if necessary.

Note:

This function will try to retrieve the folders from the cache, if possible, or fall back to the database.

Parameters:

- user* a pointer to the meta user object that will have its message folders updated.
- locked* if META_NEED_LOCK is specified, the meta user object will be locked for operation.

Returns:

-1 on failure or 1 on success.

Definition at line 310 of file updaters.c.

References `contacts_update()`, `inx_free()`, `inx_t`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `serial_get()`, and `serial_increment()`.

Referenced by `meta_get()`, and `sess_update()`.

5.221.1.3 `int_t meta_update_folders (meta_user_t * user, META_LOCK_STATUS locked)`

Update a user's message folders if necessary.

Note:

This function will try to retrieve the folders from the cache, if possible, or fall back to the database.

Parameters:

- user* a pointer to the meta user object that will have its message folders updated.
- locked* if META_NEED_LOCK is specified, the meta user object will be locked for operation.

Returns:

-1 on failure or 1 on success.

Definition at line 367 of file updaters.c.

References `meta_data_fetch_folders()`, `meta_messages_update_sequences()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_FOLDERS`, `serial_get()`, and `serial_increment()`.

Referenced by `imap_session_update()`, `meta_get()`, and `sess_update()`.

5.221.1.4 `int_t meta_update_keys (meta_user_t * user, META_LOCK_STATUS locked)`

Fetches the user signet and private key. Relies on the mail realm to decrypt the values.

Parameters:

- user* a pointer to the meta object that is to be populated.
- locked* the meta lock status of the operation (if META_NEED_LOCK is supplied, the meta user object will be locked for the duration of the function).

Returns:

-2 if there is a problem unscrambling the private key, -1 for a system error, 0 for success, and 1 if the keys were created.

Definition at line 113 of file updaters.c.

References `BINARY`, `log_pedantic`, `meta_crypto_keys_create()`, `meta_data_fetch_keys()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `NONE`, `prime_key_decrypt()`, `prime_set()`, `key_pair_t::private`, `key_pair_t::public`, `st_char_get()`, `st_cleanup`, `st_length_int()`, `st_populated`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

Referenced by `meta_get()`.

5.221.1.5 `int_t meta_update_message_folders(meta_user_t * user, META_LOCK_STATUS locked)`

Refresh a user's message folders if stale, or retrieve them from the database if they are not in memory.

See also:

[messages_update\(\)](#)

Parameters:

user a pointer to the meta user object requesting the folders.

locked if `META_LOCKED`, lock the meta user object for the duration of the request.

Returns:

-1 on failure, or 1 on success.

Definition at line 419 of file updaters.c.

References `inx_free()`, `inx_t`, `messages_update()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_FOLDERS`, `serial_get()`, and `serial_increment()`.

Referenced by `meta_get()`.

5.221.1.6 `int_t meta_update_realms(meta_user_t * user, stringer_t * salt, stringer_t * master, META_LOCK_STATUS locked)`

Fetches the user realm keys and extracts the different components.

Parameters:

user a pointer to the meta object that is to be populated.

master the user master key value.

locked the meta lock status of the operation (if `META_NEED_LOCK` is supplied, the meta user object will be locked for the duration of the function).

Returns:

-2 if there is a problem extracting the key values, -1 for a system error, 0 for success, and 1 if the keys were created.

****/

****/

Definition at line 20 of file updaters.c.

References `log_pedantic`, `MANAGEDBUF`, `meta_data_fetch_shard()`, `meta_data_insert_shard()`, `META_NEED_LOCK`, `meta_user_unlock()`, `meta_user_wlock()`, `PLACER`, `st_char_get()`, `st_empty`, `st_length_get()`, `st_length_int()`, `stacie_create_shard()`, `STACIE_KEY_LENGTH`, `stacie_realm_key()`, `STACIE_SHARD_LENGTH`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

Referenced by `meta_get()`.

5.221.1.7 `int_t meta_update_user (meta_user_t * user, META_LOCK_STATUS locked)`

Build a user's meta information from specified data parameters.

Note:

The user object will be pulled from the cache, if possible, or it falls back to a database lookup using the user number and the verification token.

Parameters:

user a pointer to the meta object that is to be populated.

usernum the user number.

verification the verification token for the specified user number.

locked the meta lock status of the operation (if META_NEED_LOCK is supplied, the meta user object will be locked for the duration of the function).

Returns:

-1 on error, 0 on success, 1 for an authentication issue.

Definition at line 211 of file updaters.c.

References `meta_data_fetch_user()`, `META_NEED_LOCK`, `meta_user_serial_get()`, `meta_user_serial_set()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_USER`, `serial_get()`, `serial_increment()`, and `st_populated`.

Referenced by `imap_session_update()`, `meta_get()`, and `sess_update()`.

5.222 src/objects/objects.h File Reference

```
#include "auth/auth.h"
#include "meta/meta.h"
#include "warehouse/warehouse.h"
#include "folders/folders.h"
#include "contacts/contacts.h"
#include "messages/messages.h"
#include "config/config.h"
#include "mail/mail.h"
#include "sessions/sessions.h"
```

Data Structures

- struct [object_cache_t](#)

Enumerations

- enum {
[OBJECT_USER](#), [OBJECT_CONFIG](#), [OBJECT_FOLDERS](#), [OBJECT_MESSAGES](#),
[OBJECT_CONTACTS](#), [OBJECT_ALIASES](#) }

Functions

- [int_t lock_get](#) ([stringer_t](#) *key)
[locks.c](#)
- void [lock_release](#) ([stringer_t](#) *key)
Release a named lock, with synchronization provided via memcached.
- [int_t user_lock](#) (uint64_t usernum)
Acquire a lock in the magma.user keyspace.
- void [user_unlock](#) (uint64_t usernum)
Unlock a lock in the magma.user keyspace.
- [bool_t obj_cache_start](#) (void)
[objects.c](#)
- void [obj_cache_prune](#) (void)
The prune function runs every few minutes and scans the object cache and removes any stale objects it finds.
- void [obj_cache_stop](#) (void)
Stop the object cache and free all active user and web session objects.
- uint64_t [serial_get](#) (uint64_t type, uint64_t num)
[serials.c](#)

- uint64_t [serial_increment](#) (uint64_t type, uint64_t num)
Increment the serial number for an object in memcached.
- uint64_t [serial_reset](#) (uint64_t type, uint64_t num)
Reset the serial number to 1 for an object in memcached.

Variables

- [object_cache_t](#) objects

5.222.1 Enumeration Type Documentation

5.222.1.1 anonymous enum

Enumerator:

OBJECT_USER
OBJECT_CONFIG
OBJECT_FOLDERS
OBJECT_MESSAGES
OBJECT_CONTACTS
OBJECT_ALIASES

Definition at line 21 of file objects.h.

5.222.2 Function Documentation

5.222.2.1 int_t lock_get (stringer_t * key)

[locks.c](#) [locks.c](#)

See also:

[cache_silent_add\(\)](#)

Note:

The lock will be held for a maximum of 10 minutes, and failed locking attempts will be retried periodically for a maximum of 1 minute before returning failure.

Parameters:

key a managed string containing the name of the lock to be acquired.

Returns:

-1 on general failure, 0 on memcached failure, or 1 on success.

Definition at line 21 of file locks.c.

References [cache_silent_add\(\)](#), [lock](#), [log_pedantic](#), [MAGMA_LOCK_EXPIRATION](#), [MAGMA_LOCK_TIMEOUT](#), [MANAGEDBUF](#), [PLACER](#), [st_char_get\(\)](#), [st_empty](#), [st_length_int\(\)](#), and [st_sprint\(\)](#).

Referenced by [api_endpoint_auth\(\)](#), [imap_login\(\)](#), [pop_pass\(\)](#), [portal_endpoint_auth\(\)](#), [smtp_auth_login\(\)](#), [smtp_auth_plain\(\)](#), [smtp_reply\(\)](#), and [user_lock\(\)](#).

5.222.2.2 void lock_release (stringer_t * *key*)

Release a named lock, with synchronization provided via memcached.

See also:

[cache_delete\(\)](#)

Note:

The lock will be held for 10 seconds, and locking attempts will occur periodically for 60 seconds prior to failure.

Parameters:

key a managed string containing the name of the lock to be released.

Returns:

-1 on general failure, 0 on memcached failure, or 1 on success.

LOW: At some point we should add logic to check whether this cluster node even owns the lock before blindly deleting the lock.

Definition at line 61 of file locks.c.

References [cache_delete\(\)](#), [lock](#), [log_pedantic](#), [MANAGEDBUF](#), [st_char_get\(\)](#), [st_empty](#), [st_length_int\(\)](#), and [st_sprint\(\)](#).

Referenced by [api_endpoint_auth\(\)](#), [imap_login\(\)](#), [pop_pass\(\)](#), [portal_endpoint_auth\(\)](#), [smtp_auth_login\(\)](#), [smtp_auth_plain\(\)](#), [smtp_reply\(\)](#), and [user_unlock\(\)](#).

5.222.2.3 void obj_cache_prune (void)

The prune function runs every few minutes and scans the object cache and removes any stale objects it finds.

Note:

If the number of entries is over 4,096, then the prune function will remove entries candidates which have been unused more than 5 minutes. If the index holds more than 2,048, entries older than 30 minutes are pruned, otherwise if the index holds fewer than 2,048 entries, only those objects older than 1 hour are removed. Also, note that the precise interval between scans is somewhat random, because the background thread responsible for running the prune function goes to sleep for a random number of seconds.

Definition at line 64 of file objects.c.

References [count](#), [inx_count\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_key_active\(\)](#), [inx_cursor_reset\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [inx_delete\(\)](#), [inx_lock_read\(\)](#), [inx_lock_write\(\)](#), [inx_unlock\(\)](#), [object_cache_t::meta](#), [meta_user_ref_stamp\(\)](#), [meta_user_ref_total\(\)](#), [meta_user_t](#), [sess_ref_stamp\(\)](#), [sess_ref_total\(\)](#), [session_t](#), [object_cache_t::sessions](#), [stats_adjust_by_name\(\)](#), and [stats_set_by_name\(\)](#).

Referenced by [process_maint\(\)](#).

5.222.2.4 bool_t obj_cache_start (void)

objects.c objects.c

Returns:

true on success or false on failure.

Definition at line 19 of file objects.c.

References [inx_alloc\(\)](#), [log_critical](#), [M_INX_LOCK_MANUAL](#), [M_INX_TREE](#), [object_cache_t::meta](#), [meta_free\(\)](#), [sess_destroy\(\)](#), and [object_cache_t::sessions](#).

Referenced by [process_start\(\)](#).

5.222.2.5 void obj_cache_stop (void)

Stop the object cache and free all active user and web session objects.

Returns:

This function returns no value.

Definition at line 38 of file objects.c.

References `inx_free()`, `object_cache_t::meta`, and `object_cache_t::sessions`.

Referenced by `process_stop()`.

5.222.2.6 uint64_t serial_get (uint64_t type, uint64_t num)

serials.c serials.c

Parameters:

type the serial type to be queried (OBJECT_USER, OBJECT_CONFIG, OBJECT_FOLDERS, OBJECT_MESSAGES, or OBJECT_CONTACTS).

num the specific object identifier.

Returns:

0 on failure or the serial number of the requested object.

Definition at line 62 of file serials.c.

References `cache_increment()`, `log_pedantic`, `serial_prefix()`, `st_aprint()`, `st_char_get()`, `st_free()`, and `st_length_int()`.

Referenced by `imap_append_message()`, `imap_close()`, `imap_create()`, `imap_delete()`, `imap_expunge()`, `imap_fetch()`, `imap_message_copier()`, `imap_rename()`, `imap_session_update()`, `imap_store()`, `meta_messages_login_update()`, `meta_messages_update()`, `meta_update_aliases()`, `meta_update_contacts()`, `meta_update_folders()`, `meta_update_message_folders()`, `meta_update_user()`, `meta_user_serial_check()`, `portal_endpoint_folders_rename()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `user_config_create()`, and `user_config_update()`.

5.222.2.7 uint64_t serial_increment (uint64_t type, uint64_t num)

Increment the serial number for an object in memcached.

Parameters:

type the serial type to be queried (OBJECT_USER, OBJECT_CONFIG, OBJECT_FOLDERS, OBJECT_MESSAGES, or OBJECT_CONTACTS).

num the specific object identifier.

Returns:

0 on failure or the new serial number of the requested object.

Definition at line 86 of file serials.c.

References `cache_increment()`, `log_pedantic`, `serial_prefix()`, `st_aprint()`, `st_char_get()`, `st_free()`, and `st_length_int()`.

Referenced by `contact_move()`, `imap_append_message()`, `imap_close()`, `imap_create()`, `imap_delete()`, `imap_expunge()`, `imap_fetch()`, `imap_message_copier()`, `imap_rename()`, `imap_store()`, `mail_load_message()`, `meta_messages_login_update()`, `meta_messages_update()`, `meta_update_aliases()`, `meta_update_contacts()`, `meta_update_folders()`, `meta_update_message_folders()`, `meta_update_user()`, `meta_user_serial_check()`, `pop_session_destroy()`, `portal_endpoint_folders_rename()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `smtp_rollout()`, `smtp_store_message()`, and `user_config_edit()`.

5.222.2.8 uint64_t serial_reset (uint64_t type, uint64_t num)

Reset the serial number to 1 for an object in memcached.

Parameters:

type the serial type to be queried (OBJECT_USER, OBJECT_CONFIG, OBJECT_FOLDERS, OBJECT_MESSAGES, or OBJECT_CONTACTS).

num the specific object identifier.

Returns:

0 on failure or the new value of the cached serial number (1) on success.

Definition at line 110 of file serials.c.

References cache_set(), CONSTANT, log_pedantic, serial_prefix(), st_aprint(), st_char_get(), st_free(), and st_length_int().

5.222.2.9 int_t user_lock (uint64_t usernum)

Acquire a lock in the magma.user keyspace.

Parameters:

usernum the numerical id of the user for whom the lock will be acquired.

Returns:

-1 on general failure, 0 on memcached failure, or 1 on success.

Definition at line 82 of file locks.c.

References lock_get(), MANAGEDBUF, and st_sprint().

Referenced by imap_close(), imap_copy(), imap_expunge(), smtp_rollout(), and smtp_store_message().

5.222.2.10 void user_unlock (uint64_t usernum)

Unlock a lock in the magma.user keyspace.

Parameters:

usernum the numerical id of the user for whom the lock will be unlocked.

Returns:

This function returns no value.

Definition at line 98 of file locks.c.

References lock_release(), MANAGEDBUF, and st_sprint().

Referenced by imap_close(), imap_copy(), imap_expunge(), smtp_rollout(), and smtp_store_message().

5.222.3 Variable Documentation**5.222.3.1 object_cache_t objects**

Definition at line 10 of file objects.c.

Referenced by __attribute__(), meta_inx_find(), meta_inx_remove(), sess_create(), and sess_get().

5.223 src/objects/sessions/sessions.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t sess_key (void)`
Generate a random 12 digit 64-bit web session key.
- `uint64_t sess_number (void)`
Reserve a unique web session identifier.
- `void sess_destroy (session_t *sess)`
Destroy a web session and its associated data.
- `void sess_ref_add (session_t *sess)`
Increment the web session's reference counter and update its timestamp.
- `void sess_ref_dec (session_t *sess)`
Decrement the web session's reference counter and update its timestamp.
- `uint64_t sess_ref_total (session_t *sess)`
Get the reference count of a web session.
- `time_t sess_ref_stamp (session_t *sess)`
Get the timestamp for a web session's reference counter.
- `void sess_refresh_flush (session_t *sess)`
Update a web session's refresh stamp and prevent redundant refreshing of a web session.
- `time_t sess_refresh_stamp (session_t *sess)`
Get the timestamp for the last time the session was refreshed.
- `bool_t sess_refresh_check (session_t *sess)`
Check to see if a web session is ready to be refreshed, and if so, disarm its trigger and update its refresh stamp.
- `stringer_t * sess_token (session_t *sess)`
Securely generate a unique zbase32-encoded token for a session.
- `void sess_update (session_t *sess)`
Update a session's underlying data and its refresh timestamp.
- `void sess_trigger (session_t *sess)`
Set a web session's trigger so it will be refreshed as soon as possible.
- `void sess_release (session_t *sess)`
Release a web session.
- `void sess_release_attachment (attachment_t *attachment)`
Free an attachment object.
- `void sess_release_composition (composition_t *comp)`

Free a composition object.

- void `sess_serial_check` (`session_t` *sess, uint64_t object)

Queue a web session refresh if there is stale data.

- `session_t` * `sess_create` (`connection_t` *con, `stringer_t` *path, `stringer_t` *application)

Create a new session for a given web connection.

- int_t `sess_get` (`connection_t` *con, `stringer_t` *application, `stringer_t` *path, `stringer_t` *token)

Try to retrieve the session associated with a client connection's supplied cookie.

Variables

- struct {
 uint64_t `number`
 pthread_mutex_t `lock`
} `sessions`

5.223.1 Function Documentation

5.223.1.1 `session_t`* `sess_create` (`connection_t` * *con*, `stringer_t` * *path*, `stringer_t` * *application*)

Create a new session for a given web connection. sessions.c

Note:

The session stores the following data points: remote IP address, request path, application name, the specified http hostname, the remote client's user agent string, the server's host number, a unique session id, the server's current timestamp, a randomly- generated session key for authentication, and an encrypted token for the session returned to the user as a cookie.

Parameters:

con a pointer to the connection underlying the web session.

path a pointer to a managed string containing the pathname of the generating request (should be "/portal/camel").

application a pointer to a managed string containing the name of the parent application of the session (should be "portal").

Returns:

NULL on failure or a pointer to a newly allocated session object for the specified connection.

Definition at line 371 of file sessions.c.

References `con_addr()`, `con_secure()`, `CONTIGUOUS`, `HEAP`, `magma_t::host`, `inx_alloc()`, `inx_insert()`, `ip_copy()`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `magma`, `MANAGED_T`, `MEMORYBUF`, `mm_alloc()`, `mm_free()`, `magma_t::number`, `objects`, `sess_destroy()`, `sess_key()`, `sess_number()`, `sess_ref_add()`, `sess_ref_dec()`, `sess_release_composition()`, `sess_token()`, `session_t`, `object_cache_t::sessions`, `st_dupe_opts()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `json_api_dispatch()`, and `portal_endpoint()`.

5.223.1.2 void `sess_destroy` (`session_t` * *sess*)

Destroy a web session and its associated data.

Parameters:

sess a pointer to the web session to be destroyed.

Returns:

This function returns no value.

Definition at line 59 of file sessions.c.

References `inx_cleanup()`, `meta_inx_remove()`, `META_PROTOCOL_WEB`, `mm_free()`, `mutex_destroy()`, and `st_cleanup`.

Referenced by `obj_cache_start()`, and `sess_create()`.

5.223.1.3 int_t sess_get (connection_t * con, stringer_t * application, stringer_t * path, stringer_t * token)

Try to retrieve the session associated with a client connection's supplied cookie.

Parameters:

con a pointer to the connection object sending the cookie.

application a managed string containing the application associated with the session.

path a managed string containing the path associated with the session.

token the encrypted user token retrieved from the supplied http cookie.

Returns:

1 if the cookie was found and valid, or one of the following values on failure: 0 = Session not found. -1 = Server error. -2 = Invalid token. -3 = Security violation / incorrect user-agent. -4 = Security violation / incorrect session key. -5 = Security violation / incorrect source address. -6 = Session terminated by logout. -7 = Session timed out.

Most session attributes need simple equality comparison, except for timeout checking. Make sure not to validate against a stale session that should have already timed out (which will have to be determined dynamically).

Definition at line 434 of file sessions.c.

References `con_addr()`, `con_secure()`, `deprecated_scramble_decrypt()`, `deprecated_scramble_import()`, `magma_t::http`, `inx_delete()`, `inx_find()`, `inx_lock_read()`, `inx_unlock()`, `ip_addr_eq()`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `magma`, `MEMORYBUF`, `number`, `objects`, `magma_t::secure`, `sess_ref_add()`, `sess_ref_dec()`, `sess_refresh_stamp()`, `sess_update()`, `object_cache_t::sessions`, `magma_t::sessions`, `st_char_get()`, `st_cleanup`, `st_cmp_cs_eq()`, `st_data_get()`, `st_free()`, `st_length_int()`, `multi_t::u64`, `multi_t::val`, and `zbase32_decode()`.

Referenced by `http_parse_context()`.

5.223.1.4 uint64_t sess_key (void)

Generate a random 12 digit 64-bit web session key.

Returns:

the randomly generated web session key as an unsigned 64 bit integer.

Definition at line 22 of file sessions.c.

References `log_pedantic`, `rand_get_uint64()`, and `uint64_digits()`.

Referenced by `sess_create()`.

5.223.1.5 uint64_t sess_number (void)

Reserve a unique web session identifier.

Returns:

a number containing a unique web session identifier.

Definition at line 43 of file sessions.c.

References mutex_lock(), mutex_unlock(), and sessions.

Referenced by sess_create().

5.223.1.6 void sess_ref_add (session_t * sess)

Increment the web session's reference counter and update its timestamp.

Parameters:

sess a pointer to the web session to be updated.

Returns:

This function returns no value.

Definition at line 87 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_create(), and sess_get().

5.223.1.7 void sess_ref_dec (session_t * sess)

Decrement the web session's reference counter and update its timestamp.

Parameters:

sess a pointer to the web session to be updated.

Returns:

This function returns no value.

Definition at line 112 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_create(), sess_get(), and sess_release().

5.223.1.8 time_t sess_ref_stamp (session_t * sess)

Get the timestamp for a web session's reference counter.

Parameters:

sess a pointer to the web session to be queried.

Returns:

the last-modified UTC timestamp value of the specified web session.

Definition at line 162 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by obj_cache_prune().

5.223.1.9 uint64_t sess_ref_total (session_t * sess)

Get the reference count of a web session.

Parameters:

sess a pointer to the web session to be queried.

Returns:

the total number of references to the specified web session.

Definition at line 137 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by obj_cache_prune().

5.223.1.10 bool_t sess_refresh_check (session_t * sess)

Check to see if a web session is ready to be refreshed, and if so, disarm its trigger and update its refresh stamp.

Note:

The caller needs to make immediate use of this function's return value because the refresh trigger will be cleared if it was previously set. Any session longer than 2 minutes will be marked as ready for refresh.

Parameters:

a pointer to the web session to be queried.

Returns:

true if the web session is ready to be refreshed, or false if it is not.

Definition at line 217 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_release().

5.223.1.11 void sess_refresh_flush (session_t * sess)

Update a web session's refresh stamp and prevent redundant refreshing of a web session.

Parameters:

sess a pointer to the web session to be touched.

Returns:

This function returns no value.

Definition at line 180 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_update().

5.223.1.12 time_t sess_refresh_stamp (session_t * sess)

Get the timestamp for the last time the session was refreshed.

Parameters:

sess a pointer to the web session to be queried.

Returns:

the last-refreshed UTC timestamp value of the specified web session.

Definition at line 197 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by sess_get().

5.223.1.13 void sess_release (session_t * sess)

Release a web session.

Note:

If the web session should have been refreshed, it will be refreshed before the reference counter is decremented.

Parameters:

sess a pointer to the web session to be released.

Returns:

This function returns no value.

Definition at line 302 of file sessions.c.

References requeue(), sess_ref_dec(), sess_refresh_check(), and sess_update().

Referenced by http_session_reset().

5.223.1.14 void sess_release_attachment (attachment_t * attachment)

Free an attachment object.

Note:

This is an inx helper function.

Parameters:

attachment a pointer to the attachment object to be destroyed.

Returns:

This function returns no value.

Definition at line 321 of file sessions.c.

References mm_free(), and st_cleanup.

Referenced by portal_endpoint_attachments_add(), and portal_endpoint_messages_compose().

5.223.1.15 void sess_release_composition (composition_t * comp)

Free a composition object.

Note:

This is an inx helper function.

Parameters:

comp a pointer to the composition object to be destroyed.

Returns:

This function returns no value.

Definition at line 337 of file sessions.c.

References inx_cleanup(), and mm_free().

Referenced by portal_endpoint_messages_compose(), and sess_create().

5.223.1.16 void sess_serial_check (session_t * sess, uint64_t object)

Queue a web session refresh if there is stale data.

Parameters:

sess a pointer to the web session to be queried.

object the value of the object in the cache to be checked (can include OBJECT_CONTACTS and OBJECT_FOLDERS).

Definition at line 352 of file sessions.c.

References meta_user_serial_check(), and sess_trigger().

Referenced by portal_endpoint_contacts_add(), portal_endpoint_contacts_copy(), portal_endpoint_contacts_edit(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), portal_folder_contacts_add(), portal_folder_contacts_remove(), portal_folder_mail_add(), and portal_folder_mail_remove().

5.223.1.17 stringer_t* sess_token (session_t * sess)

Securely generate a unique zbase32-encoded token for a session.

Parameters:

sess a pointer to the input web session.

Returns:

a managed string containing the generated web session token.

Definition at line 241 of file sessions.c.

References deprecated_scramble_encrypt(), deprecated_scramble_free(), deprecated_scramble_total_length(), log_pedantic, magma, PLACER, magma_t::secure, magma_t::sessions, st_cleanup, st_merge, and zbase32_encode().

Referenced by sess_create().

5.223.1.18 void sess_trigger (session_t * sess)

Set a web session's trigger so it will be refreshed as soon as possible.

Parameters:

sess a pointer to the web session to be refreshed.

Returns:

This function returns no value.

Definition at line 285 of file sessions.c.

References mutex_lock(), and mutex_unlock().

Referenced by portal_endpoint_folders_rename(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), and sess_serial_check().

5.223.1.19 void sess_update (session_t * sess)

Update a session's underlying data and its refresh timestamp.

Note:

This function ensures the session's associated meta user data, mail folders and messages, and contact folders and contact entries are all current.

Parameters:

sess a pointer to the session object to be updated.

Returns:

This function returns no value.

Definition at line 264 of file sessions.c.

References meta_messages_update(), META_NEED_LOCK, meta_update_aliases(), meta_update_contacts(), meta_update_folders(), meta_update_user(), and sess_refresh_flush().

Referenced by sess_get(), and sess_release().

5.223.2 Variable Documentation**5.223.2.1 pthread_mutex_t lock**

Definition at line 12 of file sessions.c.

5.223.2.2 uint64_t number

Definition at line 11 of file sessions.c.

Referenced by __attribute__(), deprecated_ecies_key_private(), ecies_key_private(), hex_encode_chr(), imap_build_array(), imap_fetch_body(), imap_fetch_body_header(), imap_fetch_body_part(), imap_fetch_body_tag(), imap_fetch_parse_partial(), imap_flag_parse(), imap_narrow_messages(), imap_parse_dataitems(), imap_parse_literal(), imap_search_messages_inner(), imap_search_messages_range(), imap_status(), pop_dele(), pop_last(), pop_list(), pop_retr(), pop_top(), pop_uidl(), secp256k1_private_set(), sess_get(), smtp_bounce(), and smtp_check_receive_quota().

5.223.2.3 struct { ... } sessions

Referenced by sess_number().

5.224 src/servers/http/sessions.c File Reference

```
#include "magma.h"
```

Functions

- void [http_session_reset](#) ([connection_t](#) *con)
Reset a client http connection to its original, uninitialized state.
- void [http_session_destroy](#) ([connection_t](#) *con)
Destroy an http client connection.

5.224.1 Function Documentation

5.224.1.1 void [http_session_destroy](#) ([connection_t](#) * con)

Destroy an http client connection. sessions.c

Parameters:

con the connection object to have its http session destroyed.

Returns:

This function returns no value.

Definition at line 77 of file sessions.c.

References [http_session_reset](#)().

Referenced by [con_destroy](#)().

5.224.1.2 void [http_session_reset](#) ([connection_t](#) * con)

Reset a client http connection to its original, uninitialized state.

Parameters:

con the connection object to have its http session state reset.

Returns:

This function returns no value.

Definition at line 15 of file sessions.c.

References [HTTP_CLOSE](#), [HTTP_CONNECTION_CLOSE](#), [HTTP_CONNECTION_NEUTRAL](#), [HTTP_MERGED](#), [HTTP_METHOD_NONE](#), [HTTP_PORTAL](#), [HTTP_READY](#), [inx_cleanup](#)(), [json_decref_d](#), [sess_release](#)(), and [st_cleanup](#).

Referenced by [http_requeue](#)(), and [http_session_destroy](#)().

5.225 src/servers/imap/sessions.c File Reference

```
#include "magma.h"
```

Functions

- [int_t imap_session_update](#) ([connection_t](#) *con)

Returns 0 if the selected folder wasn't modified, or 1 if things changed and the updated status should be sent to the client, and a -1 is used to indicate the update check encountered a problem and should be retried later.

- void [imap_session_destroy](#) ([connection_t](#) *con)

sessions.c

5.225.1 Function Documentation

5.225.1.1 void [imap_session_destroy](#) ([connection_t](#) * con)

sessions.c

Definition at line 93 of file sessions.c.

References [ar_free\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [mail_cache_reset\(\)](#), [MAIL_STATUS_RECENT](#), [meta_inx_remove\(\)](#), [meta_message_t](#), [META_PROTOCOL_IMAP](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), and [st_cleanup](#).

Referenced by [con_destroy\(\)](#).

5.225.1.2 int_t [imap_session_update](#) ([connection_t](#) * con)

Returns 0 if the selected folder wasn't modified, or 1 if things changed and the updated status should be sent to the client, and a -1 is used to indicate the update check encountered a problem and should be retried later.

Definition at line 14 of file sessions.c.

References [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [MAIL_STATUS_RECENT](#), [META_LOCKED](#), [meta_message_t](#), [meta_messages_update\(\)](#), [meta_update_folders\(\)](#), [meta_update_user\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_FOLDERS](#), [OBJECT_MESSAGES](#), [OBJECT_USER](#), and [serial_get\(\)](#).

Referenced by [imap_check\(\)](#), [imap_expunge\(\)](#), [imap_noop\(\)](#), and [imap_store\(\)](#).

5.226 src/servers/molten/sessions.c File Reference

```
#include "magma.h"
```

Functions

- void [molten_session_destroy](#) ([connection_t](#) *con)
sessions.c

5.226.1 Function Documentation

5.226.1.1 void molten_session_destroy (connection_t * con)

sessions.c

Definition at line 10 of file sessions.c.

Referenced by [con_destroy\(\)](#).

5.227 src/servers/pop/sessions.c File Reference

```
#include "magma.h"
```

Functions

- [int_t pop_session_reset](#) ([connection_t](#) *con)
Reset a POP3 session by guaranteeing that no messages are flagged to be deleted.
- [void pop_session_destroy](#) ([connection_t](#) *con)
Destroy a POP3 session and delete any flagged messages belonging to the user.

5.227.1 Function Documentation

5.227.1.1 void pop_session_destroy (connection_t * con)

Destroy a POP3 session and delete any flagged messages belonging to the user. sessions.c

Parameters:

con the POP3 client connection issuing the command. This function returns no value.

Definition at line 50 of file sessions.c.

References [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [inx_delete\(\)](#), [M_TYPE_UINT64](#), [mail_cache_reset\(\)](#), [mail_remove_message\(\)](#), [MAIL_STATUS_HIDDEN](#), [meta_inx_remove\(\)](#), [meta_message_t](#), [meta_messages_update_sequences\(\)](#), [META_PROTOCOL_POP](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_MESSAGES](#), [serial_increment\(\)](#), [st_cleanup](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [con_destroy\(\)](#).

5.227.1.2 int_t pop_session_reset (connection_t * con)

Reset a POP3 session by guaranteeing that no messages are flagged to be deleted.

Parameters:

con the POP3 client connection issuing the command.

Returns:

-1 on general error, 0 if the connection hasn't been authenticated, or 1 if all messages have had their statuses successfully reset.

Definition at line 15 of file sessions.c.

References [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [MAIL_STATUS_HIDDEN](#), [meta_message_t](#), [meta_user_unlock\(\)](#), and [meta_user_wlock\(\)](#).

Referenced by [pop_rset\(\)](#), and [pop_starttls\(\)](#).

5.228 src/web/register/sessions.c File Reference

```
#include "magma.h"
```

Functions

- void [register_session_free](#) ([register_session_t](#) *session)
Destroy a registration session and all its associated data.
- [register_session_t](#) * [register_session_generate](#) (void)
Generate a new registration session.
- [register_session_t](#) * [register_session_get](#) ([connection_t](#) *con, [stringer_t](#) *name)
Retrieve a registration session by a data key supplied in the user's http POST request.
- [bool_t](#) [register_session_cache](#) ([connection_t](#) *con, [register_session_t](#) *session)
Save a registration session's state into the cache.

5.228.1 Function Documentation

5.228.1.1 [bool_t](#) [register_session_cache](#) ([connection_t](#) * con, [register_session_t](#) * session)

Save a registration session's state into the cache. sessions.c

Note:

The session is stored under the parent key "lavad.register.session."

Parameters:

con a pointer to the client connection underlying the user session.
session a pointer to the registration session to be persisted.

Returns:

false on failure or true if the session was cached successfully.

Definition at line 105 of file sessions.c.

References [cache_set\(\)](#), [con_addr_presentation\(\)](#), [data](#), [register_session_t::hvf_input](#), [register_session_t::hvf_value](#), [log_pedantic](#), [MAN-AGEDBUF](#), [register_session_t::name](#), [NULLER](#), [register_session_t::password](#), [register_session_t::plan](#), [serialize_st\(\)](#), [serialize_uint16\(\)](#), [serialize_uint64\(\)](#), [st_char_get\(\)](#), [st_cleanup](#), [st_free\(\)](#), [st_length_int\(\)](#), [register_session_t::username](#), and [register_session_t::usernum](#).

Referenced by [register_process\(\)](#).

5.228.1.2 [void](#) [register_session_free](#) ([register_session_t](#) * session)

Destroy a registration session and all its associated data.

Returns:

This function returns no value.

Definition at line 14 of file sessions.c.

References `register_session_t::hvf_input`, `register_session_t::hvf_value`, `mm_free()`, `register_session_t::name`, `register_session_t::password`, `st_cleanup`, and `register_session_t::username`.

Referenced by `register_process()`, `register_session_generate()`, and `register_session_get()`.

5.228.1.3 `register_session_t*` `register_session_generate` (`void`)

Generate a new registration session.

Returns:

NULL on failure, or a pointer to a new randomly named session on success.

Definition at line 35 of file sessions.c.

References `log_pedantic`, `mm_alloc()`, `register_session_t::name`, `rand_choices()`, and `register_session_free()`.

Referenced by `register_process()`.

5.228.1.4 `register_session_t*` `register_session_get` (`connection_t * con`, `stringer_t * name`)

Retrieve a registration session by a data key supplied in the user's http POST request.

Note:

The session is stored under the parent key "lavad.register.session."

Parameters:

con the client connection underlying the user request.

name a managed string containing the registration session identifier.

Returns:

NULL on failure, or a pointer to the user's registration session on success.

Definition at line 59 of file sessions.c.

References `cache_get()`, `con_addr_presentation()`, `data`, `deserialize_st()`, `deserialize_uint16()`, `deserialize_uint64()`, `register_session_t::hvf_input`, `register_session_t::hvf_value`, `log_pedantic`, `MANAGEDBUF`, `mm_alloc()`, `mm_wipe()`, `register_session_t::name`, `NULLER`, `register_session_t::password`, `register_session_t::plan`, `register_session_free()`, `serialization_t`, `st_char_get()`, `st_dupe()`, `st_free()`, `st_length_int()`, `register_session_t::username`, and `register_session_t::usernum`.

Referenced by `register_process()`.

5.229 src/objects/warehouse/domains.c File Reference

```
#include "magma.h"
```

Functions

- void `domain_stop` (void)
Free the global list of domains.
- `bool_t domain_start` (void)
Fetch and store the list of configured domains from the database.
- `domain_t * domain_alloc` (`stringer_t *domain`, `int_t restricted`, `int_t mailboxes`, `int_t wildcard`, `int_t dkim`, `int_t spf`)
Allocate and initialize a new domain object.
- `int_t domain_mailboxes` (`stringer_t *domain`)
Determine whether mailboxes are hosted locally for a specified domain.
- `int_t domain_restricted` (`stringer_t *domain`)
Determine whether a specified domain is restricted to authenticated users.
- `int_t domain_wildcard` (`stringer_t *domain`)
Determine whether a specified domain has wildcards enabled.
- `int_t domain_dkim` (`stringer_t *domain`)
TODO: Eliminate dkim+spf and replace them with a 'sign' flag.
- `int_t domain_spf` (`stringer_t *domain`)
Determine whether SPF has been configured for a specified domain.

Variables

- `inx_t * domains` = NULL

5.229.1 Function Documentation

5.229.1.1 `domain_t* domain_alloc (stringer_t * domain, int_t restricted, int_t mailboxes, int_t wildcard, int_t dkim, int_t spf)`

Allocate and initialize a new domain object. [domains.c](#)

Parameters:

domain a pointer to a managed string containing the name of the specified domain.
restricted if set, indicate that the domain is restricted to authenticated users only.
mailboxes if set, indicate that mailboxes are hosted locally for the domain.
wildcard if set, indicate that wildcards are enabled for the domain.
dkim if set, indicate that outbound messages for the domain should be signed via DKIM.
spf if set, indicate that SPF is configured for the domain.

Returns:

NULL on failure or a pointer to the newly initialized domain object on success.

Definition at line 47 of file domains.c.

References align(), domain_t, FOREIGNDATA, JOINTED, log_pedantic, mm_alloc(), mm_copy(), PLACER_T, placer_t, st_data_get(), st_length_get(), and STACK.

Referenced by warehouse_fetch_domains().

5.229.1.2 int_t domain_dkim (stringer_t * domain)

TODO: Eliminate dkim+spf and replace them with a 'sign' flag. Determine whether outbound messages should be signed via DKIM for a specified domain.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain wasn't found, 0 if DKIM signing is disabled, or 1 if outbound messages should be signed via DKIM for the domain.

Definition at line 132 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

Referenced by mail_add_outbound_headers().

5.229.1.3 int_t domain_mailboxes (stringer_t * domain)

Determine whether mailboxes are hosted locally for a specified domain.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain was not found, 0 if the domain is foreign, or 1 if the mailbox is hosted locally for the domain.

Definition at line 76 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

5.229.1.4 int_t domain_restricted (stringer_t * domain)

Determine whether a specified domain is restricted to authenticated users.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain was not found, 0 if restricted relay is disabled, or 1 if the domain is restricted to authenticated users only.

Definition at line 94 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

5.229.1.5 `int_t domain_spf (stringer_t * domain)`

Determine whether SPF has been configured for a specified domain.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain wasn't found, 0 if SPF is disabled, or 1 if SPF is actively configured for the domain.

Definition at line 150 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

5.229.1.6 `bool_t domain_start (void)`

Fetch and store the list of configured domains from the database.

Returns:

true if the domain retrieval was successful, or false on error.

Definition at line 28 of file domains.c.

References domains, and warehouse_fetch_domains().

Referenced by warehouse_start().

5.229.1.7 `void domain_stop (void)`

Free the global list of domains.

Returns:

This function returns no value.

Definition at line 16 of file domains.c.

References domains, and inx_cleanup().

Referenced by warehouse_stop().

5.229.1.8 `int_t domain_wildcard (stringer_t * domain)`

Determine whether a specified domain has wildcards enabled.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain wasn't found, 0 if wildcards are disabled, or 1 if the domain has wildcards enabled.

Definition at line 112 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

Referenced by smtp_check_authorized_from(), and smtp_fetch_inbound().

5.229.2 Variable Documentation

5.229.2.1 `inx_t* domains = NULL`

Definition at line 10 of file domains.c.

Referenced by `domain_dkim()`, `domain_mailboxes()`, `domain_restricted()`, `domain_spf()`, `domain_start()`, `domain_stop()`, and `domain-wildcard()`.

5.230 src/objects/warehouse/patterns.c File Reference

```
#include "magma.h"
```

Functions

- `int_t pattern_check (stringer_t *message)`
Check to see if any of the entries in the patterns list are found in a body of text.
- `void pattern_update (void)`
Update the patterns list from the database, but no more frequently than once daily.
- `void pattern_stop (void)`
Destroy the patterns list.
- `bool_t pattern_start (void)`
Initialize the patterns list.

Variables

- `inx_t * patterns_list = NULL`
- `uint64_t patterns_stamp = 0`
- `pthread_mutex_t patterns_mutex = PTHREAD_MUTEX_INITIALIZER`

5.230.1 Function Documentation

5.230.1.1 `int_t pattern_check (stringer_t * message)`

Check to see if any of the entries in the patterns list are found in a body of text. [patterns.c](#)

Parameters:

message a managed string containing the raw data of the text to be searched.

Returns:

-2 on pattern match, -1 if an error occurs, or 1 if none of the patterns in the patterns list were detected.

Definition at line 19 of file `patterns.c`.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `mutex_lock()`, `mutex_unlock()`, `patterns_list`, `patterns_mutex`, `st_search_ci()`, and `stats_adjust_by_name()`.

Referenced by `portal_outbound_checks()`, and `smtp_data_outbound()`.

5.230.1.2 `bool_t pattern_start (void)`

Initialize the patterns list.

Returns:

This function will always return true.

Definition at line 109 of file patterns.c.

References pattern_update().

Referenced by warehouse_start().

5.230.1.3 void pattern_stop (void)

Destroy the patterns list.

Returns:

This function returns no value.

Definition at line 94 of file patterns.c.

References inx_cleanup(), mutex_lock(), mutex_unlock(), patterns_list, and patterns_mutex.

Referenced by warehouse_stop().

5.230.1.4 void pattern_update (void)

Update the patterns list from the database, but no more frequently than once daily.

Returns:

This function returns no value.

Definition at line 62 of file patterns.c.

References inx_cleanup(), inx_t, mutex_lock(), mutex_unlock(), patterns_list, patterns_mutex, patterns_stamp, time_datestamp(), and warehouse_fetch_patterns().

Referenced by pattern_start(), and warehouse_update().

5.230.2 Variable Documentation

5.230.2.1 inx_t* patterns_list = NULL

Definition at line 10 of file patterns.c.

Referenced by pattern_check(), pattern_stop(), and pattern_update().

5.230.2.2 pthread_mutex_t patterns_mutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 12 of file patterns.c.

Referenced by pattern_check(), pattern_stop(), and pattern_update().

5.230.2.3 uint64_t patterns_stamp = 0

Definition at line 11 of file patterns.c.

Referenced by pattern_update().

5.231 src/objects/warehouse/warehouse.c File Reference

```
#include "magma.h"
```

Functions

- void [warehouse_update](#) (void)
Update the warehouse components.
- void [warehouse_stop](#) (void)
Stop the warehouse facility.
- [bool_t](#) [warehouse_start](#) (void)
Start the warehouse facility.

5.231.1 Function Documentation

5.231.1.1 [bool_t](#) [warehouse_start](#) (void)

Start the warehouse facility. [warehouse.c](#)

Note:

This will initialize the domains and patterns lists.

Returns:

false if any of the warehouse components failed to load, or true on success.

Definition at line 40 of file [warehouse.c](#).

References [domain_start\(\)](#), [pattern_start\(\)](#), and [warehouse_stop\(\)](#).

Referenced by [process_start\(\)](#).

5.231.1.2 [void](#) [warehouse_stop](#) (void)

Stop the warehouse facility.

Note:

This will destroy the patterns and domain lists.

Returns:

This function returns no value.

Definition at line 27 of file [warehouse.c](#).

References [domain_stop\(\)](#), and [pattern_stop\(\)](#).

Referenced by [process_stop\(\)](#), and [warehouse_start\(\)](#).

5.231.1.3 void warehouse_update (void)

Update the warehouse components.

Note:

This will update the patterns list.

Returns:

This function returns no value.

Definition at line 15 of file warehouse.c.

References `pattern_update()`.

Referenced by `process_maint()`.

5.232 src/objects/warehouse/warehouse.h File Reference

Functions

- struct `__attribute__((packed))`
- `inx_t * warehouse_fetch_domains` (void)
datatier.c
- `inx_t * warehouse_fetch_patterns` (void)
Fetch the pattern list from the database.
- `domain_t * domain_alloc` (`stringer_t *domain`, `int_t restricted`, `int_t mailboxes`, `int_t wildcard`, `int_t dkim`, `int_t spf`)
domains.c
- `int_t domain_dkim` (`stringer_t *domain`)
TODO: Eliminate dkim+spf and replace them with a 'sign' flag.
- `int_t domain_mailboxes` (`stringer_t *domain`)
Determine whether mailboxes are hosted locally for a specified domain.
- `int_t domain_restricted` (`stringer_t *domain`)
Determine whether a specified domain is restricted to authenticated users.
- `int_t domain_spf` (`stringer_t *domain`)
Determine whether SPF has been configured for a specified domain.
- `bool_t domain_start` (void)
Fetch and store the list of configured domains from the database.
- `void domain_stop` (void)
Free the global list of domains.
- `int_t domain_wildcard` (`stringer_t *domain`)
Determine whether a specified domain has wildcards enabled.
- `int_t pattern_check` (`stringer_t *message`)
patterns.c
- `bool_t pattern_start` (void)
Initialize the patterns list.
- `void pattern_stop` (void)
Destroy the patterns list.
- `void pattern_update` (void)
Update the patterns list from the database, but no more frequently than once daily.
- `bool_t warehouse_start` (void)
warehouse.c
- `void warehouse_stop` (void)
Stop the warehouse facility.

- void [warehouse_update](#) (void)
Update the warehouse components.

Variables

- [domain_t](#)

5.232.1 Function Documentation

5.232.1.1 struct __attribute__((packed)) [read]

HIGH: Create an interface for loading SSL/TLS and DKIM certificates from the database. HIGH: Create an interface for checking whether an address or IP is trusted.

Definition at line 14 of file warehouse.h.

5.232.1.2 domain_t* domain_alloc (stringer_t * domain, int_t restricted, int_t mailboxes, int_t wildcard, int_t dkim, int_t spf)

[domains.c](#) [domains.c](#)

Parameters:

- domain* a pointer to a managed string containing the name of the specified domain.
- restricted* if set, indicate that the domain is restricted to authenticated users only.
- mailboxes* if set, indicate that mailboxes are hosted locally for the domain.
- wildcard* if set, indicate that wildcards are enabled for the domain.
- dkim* if set, indicate that outbound messages for the domain should be signed via DKIM.
- spf* if set, indicate that SPF is configured for the domain.

Returns:

NULL on failure or a pointer to the newly initialized domain object on success.

Definition at line 47 of file domains.c.

References [align\(\)](#), [domain_t](#), [FOREIGNDATA](#), [JOINTED](#), [log_pedantic](#), [mm_alloc\(\)](#), [mm_copy\(\)](#), [PLACER_T](#), [placer_t](#), [st_data_get\(\)](#), [st_length_get\(\)](#), and [STACK](#).

Referenced by [warehouse_fetch_domains\(\)](#).

5.232.1.3 int_t domain_dkim (stringer_t * domain)

TODO: Eliminate dkim+spf and replace them with a 'sign' flag. Determine whether outbound messages should be signed via DKIM for a specified domain.

Parameters:

- domain* a pointer to a managed string containing the name of the domain to be queried.

Returns:

- 1 on failure or if the domain wasn't found, 0 if DKIM signing is disabled, or 1 if outbound messages should be signed via DKIM for the domain.

Definition at line 132 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

Referenced by mail_add_outbound_headers().

5.232.1.4 int_t domain_mailboxes (stringer_t * domain)

Determine whether mailboxes are hosted locally for a specified domain.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain was not found, 0 if the domain is foreign, or 1 if the mailbox is hosted locally for the domain.

Definition at line 76 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

5.232.1.5 int_t domain_restricted (stringer_t * domain)

Determine whether a specified domain is restricted to authenticated users.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain was not found, 0 if restricted relay is disabled, or 1 if the domain is restricted to authenticated users only.

Definition at line 94 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

5.232.1.6 int_t domain_spf (stringer_t * domain)

Determine whether SPF has been configured for a specified domain.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain wasn't found, 0 if SPF is disabled, or 1 if SPF is actively configured for the domain.

Definition at line 150 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

5.232.1.7 bool_t domain_start (void)

Fetch and store the list of configured domains from the database.

Returns:

true if the domain retrieval was successful, or false on error.

Definition at line 28 of file domains.c.

References domains, and warehouse_fetch_domains().

Referenced by warehouse_start().

5.232.1.8 void domain_stop (void)

Free the global list of domains.

Returns:

This function returns no value.

Definition at line 16 of file domains.c.

References domains, and inx_cleanup().

Referenced by warehouse_stop().

5.232.1.9 int_t domain_wildcard (stringer_t * domain)

Determine whether a specified domain has wildcards enabled.

Parameters:

domain a pointer to a managed string containing the name of the domain to be queried.

Returns:

-1 on failure or if the domain wasn't found, 0 if wildcards are disabled, or 1 if the domain has wildcards enabled.

Definition at line 112 of file domains.c.

References domain_t, domains, inx_find(), and M_TYPE_STRINGER.

Referenced by smtp_check_authorized_from(), and smtp_fetch_inbound().

5.232.1.10 int_t pattern_check (stringer_t * message)

[patterns.c](#) [patterns.c](#)

Parameters:

message a managed string containing the raw data of the text to be searched.

Returns:

-2 on pattern match, -1 if an error occurs, or 1 if none of the patterns in the patterns list were detected.

Definition at line 19 of file patterns.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), mutex_lock(), mutex_unlock(), patterns_list, patterns_mutex, st_search_ci(), and stats_adjust_by_name().

Referenced by portal_outbound_checks(), and smtp_data_outbound().

5.232.1.11 bool_t pattern_start (void)

Initialize the patterns list.

Returns:

This function will always return true.

Definition at line 109 of file patterns.c.

References pattern_update().

Referenced by warehouse_start().

5.232.1.12 void pattern_stop (void)

Destroy the patterns list.

Returns:

This function returns no value.

Definition at line 94 of file patterns.c.

References inx_cleanup(), mutex_lock(), mutex_unlock(), patterns_list, and patterns_mutex.

Referenced by warehouse_stop().

5.232.1.13 void pattern_update (void)

Update the patterns list from the database, but no more frequently than once daily.

Returns:

This function returns no value.

Definition at line 62 of file patterns.c.

References inx_cleanup(), inx_t, mutex_lock(), mutex_unlock(), patterns_list, patterns_mutex, patterns_stamp, time_datestamp(), and warehouse_fetch_patterns().

Referenced by pattern_start(), and warehouse_update().

5.232.1.14 inx_t* warehouse_fetch_domains (void)

datatier.c datatier.c

Returns:

NULL on failure, or an inx object holding all the configured domains on success.

Definition at line 14 of file datatier.c.

References domain_alloc(), domain_t, inx_alloc(), inx_free(), inx_insert(), inx_t, log_check, log_info, log_pedantic, M_INX_TREE, M_TYPE_STRINGER, MAGMA_HOSTNAME_MAX, mm_free(), PLACER, res_field_block(), res_field_int8(), res_field_length(), res_row_next(), res_table_free(), multi_t::st, stmt_get_result(), and multi_t::val.

Referenced by domain_start().

5.232.1.15 inx_t* warehouse_fetch_patterns (void)

Fetch the pattern list from the database.

Returns:

NULL on failure or a pointer to an inx holder containing a collection of managed strings with the patterns on success.

Definition at line 64 of file `datatier.c`.

References `inx_alloc()`, `inx_free()`, `inx_insert()`, `inx_t`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `res_field_string()`, `res_row_next()`, `res_table_free()`, `st_free()`, `stmt_get_result()`, `multi_t::u64`, and `multi_t::val`.

Referenced by `pattern_update()`.

5.232.1.16 `bool_t warehouse_start (void)`

[warehouse.c](#) [warehouse.c](#)

Note:

This will initialize the domains and patterns lists.

Returns:

false if any of the warehouse components failed to load, or true on success.

Definition at line 40 of file `warehouse.c`.

References `domain_start()`, `pattern_start()`, and `warehouse_stop()`.

Referenced by `process_start()`.

5.232.1.17 `void warehouse_stop (void)`

Stop the warehouse facility.

Note:

This will destroy the patterns and domain lists.

Returns:

This function returns no value.

Definition at line 27 of file `warehouse.c`.

References `domain_stop()`, and `pattern_stop()`.

Referenced by `process_stop()`, and `warehouse_start()`.

5.232.1.18 `void warehouse_update (void)`

Update the warehouse components.

Note:

This will update the patterns list.

Returns:

This function returns no value.

Definition at line 15 of file `warehouse.c`.

References `pattern_update()`.

Referenced by `process_maint()`.

5.232.2 Variable Documentation

5.232.2.1 domain_t

Definition at line 21 of file warehouse.h.

Referenced by domain_alloc(), domain_dkim(), domain_mailboxes(), domain_restricted(), domain_spf(), domain_wildcard(), and warehouse_fetch_domains().

5.233 src/providers/checkers/allocations.h File Reference

Variables

- struct {
 uint32_t status
 uint32_t owner
 uint32_t weight
 chr_t * comment
} ip_v4_allocations_t []

5.233.1 Variable Documentation

5.233.1.1 chr_t* comment

Definition at line 30 of file allocations.h.

Referenced by nvp_t::__attribute__(), and mail_extract_address().

5.233.1.2 struct { ... } ip_v4_allocations_t []

5.233.1.3 uint32_t owner

Definition at line 29 of file allocations.h.

5.233.1.4 uint32_t status

Definition at line 29 of file allocations.h.

Referenced by __attribute__(), _do_ocsp_validation(), _sgnt_resolv_dmtip_verify_signet(), client_connect(), client_read(), client_read_line(), client_write(), con_read(), con_read_line(), con_reverse_check(), con_write_bl(), dequeue(), dh_exchange_2048(), dh_exchange_4096(), dh_params_generate(), dh_params_generate_callback(), dkim_signature_create(), dkim_signature_verify(), dmtip_requeue(), http_requeue(), imap_examine(), imap_fetch(), imap_logout(), imap_message_copier(), imap_parse_address_breaker(), imap_requeue(), imap_search(), imap_search_messages(), imap_select(), imap_status(), mail_add_forward_headers(), meta_messages_copier(), net_accept(), pop_requeue(), portal_endpoint_config_edit(), portal_endpoint_contacts_add(), portal_endpoint_contacts_edit(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), process_maint(), process_stop(), protocol_secure(), queue_signal(), signal_status(), smtp_data_finish(), smtp_data_read(), smtp_requeue(), smtp_store_message(), stacie_derive_key(), tcp_continue(), tls_continue(), tls_print(), and tls_server_alloc().

5.233.1.5 uint32_t weight

Definition at line 29 of file allocations.h.

5.234 src/providers/checkers/checkers.h File Reference

Defines

- #define `IP_RANDOMIZER_POOL` 1024
- #define `IP_RANDOMIZER_PUSH_MIN` 4
- #define `IP_RANDOMIZER_PUSH_MAX` 16

Enumerations

- enum { `UNALLOCATED` = 0, `ALLOCATED` = 1, `RESERVED` = 2 }
- enum { `REGISTRY` = 1, `DIRECT` = 2 }
- enum { `DOMESTIC` = 0, `FOREIGN` = 1 }

Functions

- `bool_t lib_load_clamav` (void)
clamav.c
- `bool_t virus_start` (void)
- `const char * lib_version_clamav` (void)
- `int virus_engine_refresh` (void)
- `int virus_check` (`stringer_t *data`)
Virus scan a block of data.
- `struct cl_engine * virus_engine_create` (`uint64_t *signatures`)
- `uint64_t virus_sigs_loaded` (void)
Get the number of virus signatures loaded by the ClamAV engine context.
- `uint64_t virus_sigs_total` (void)
Get the number of official signatures available inside the ClamAV signature directory.
- `void virus_engine_destroy` (`struct cl_engine **target`)
Destroy a ClamAV engine context.
- `void virus_stop` (void)
Shut down the ClamAV library and free all its data and temporary files.
- `int_t dkim_signature_verify` (`stringer_t *id`, `stringer_t *message`)
dkim.c
- `stringer_t * dkim_signature_create` (`stringer_t *id`, `stringer_t *domain`, `stringer_t *message`)
Generate a DKIM signature for a message.
- `void * dkim_memory_alloc` (`void *closure`, `size_t nbytes`)
A memory allocation routine stub for dkim.
- `void dkim_memory_free` (`void *closure`, `void *ptr`)
A memory free routine stub for dkim.
- `bool_t dkim_start` (void)
Start the dkim engine.

- void [dkim_stop](#) (void)
Stop the dkim engine.
- [bool_t lib_load_dkim](#) (void)
Initialize the dkim library and bind dynamically to the exported functions that are required.
- const [chr_t * lib_version_dkim](#) (void)
Return the version string of the dkim library.
- [int_t dspam_check](#) (uint64_t usernum, [stringer_t](#) *message, [stringer_t](#) **signature)
dspam.c
- [bool_t dspam_start](#) (void)
- void [dspam_stop](#) (void)
- [bool_t dspam_train](#) (uint64_t usernum, [int_t](#) disposition, [stringer_t](#) *signature)
- [bool_t lib_load_dspam](#) (void)
Initialize the dspam library and bind dynamically to the exported functions that are required.
- [chr_t * lib_version_dspam](#) (void)
Return the version string of the dspam library.
- [bool_t lib_load_spf](#) (void)
spf.c
- const [chr_t * lib_version_spf](#) (void)
Return the version string of the spf library.
- [bool_t spf_start](#) (void)
Initialize the pool of spf server connections.
- [int_t spf_check](#) (void *ip, [stringer_t](#) *helo, [stringer_t](#) *mailfrom)
Validate an smtp request via spf.
- void [spf_stop](#) (void)
Destroy the spf connection pool.

5.234.1 Define Documentation

5.234.1.1 #define IP_RANDOMIZER_POOL 1024

Definition at line 11 of file checkers.h.

5.234.1.2 #define IP_RANDOMIZER_PUSH_MAX 16

Definition at line 14 of file checkers.h.

5.234.1.3 #define IP_RANDOMIZER_PUSH_MIN 4

Definition at line 13 of file checkers.h.

5.234.2 Enumeration Type Documentation

5.234.2.1 anonymous enum

Enumerator:

UNALLOCATED

ALLOCATED

RESERVED

Definition at line 16 of file checkers.h.

5.234.2.2 anonymous enum

Enumerator:

REGISTRY

DIRECT

Definition at line 22 of file checkers.h.

5.234.2.3 anonymous enum

Enumerator:

DOMESTIC

FOREIGN

Definition at line 27 of file checkers.h.

5.234.3 Function Documentation

5.234.3.1 void* dkim_memory_alloc (void * *closure*, size_t *nbytes*)

A memory allocation routine stub for dkim.

Note:

This function is passed to dkim_init().

Parameters:

closure a void pointer to a memory closure passed to dkim_sign() and dkim_verify().

nbytes the size, in bytes, of the memory block to be allocated.

Returns:

a pointer to a newly allocated block of memory of specified size.

Definition at line 59 of file dkim.c.

References mm_alloc().

Referenced by dkim_start().

5.234.3.2 void dkim_memory_free (void * *closure*, void * *ptr*)

A memory free routine stub for dkim.

Note:

This function is passed to dkim_init().

Parameters:

closure a void pointer to a memory closure passed to dkim_sign() and dkim_verify().

ptr a pointer to the block of memory to be released.

Returns:

This function returns no value.

Definition at line 71 of file dkim.c.

References mm_free().

Referenced by dkim_start().

5.234.3.3 stringer_t* dkim_signature_create (stringer_t * *id*, stringer_t * *domain*, stringer_t * *message*)

Generate a DKIM signature for a message.

Note:

For this function to work, the [magma.dkim.enabled](#) configuration option must be set. This function also updates the provider.dkim.signed statistic.

Parameters:

id a managed string containing a printable string id for this message.

domain the domain name hosting the identified DKIM public key.

message a managed string containing the mail message data.

Returns:

NULL on failure or if the global [magma.dkim.enabled](#) setting is false; otherwise, a managed string containing a DKIM-Signature header built using the dkim signature that was generated for the input message.

Definition at line 141 of file dkim.c.

References CONTIGUOUS, magma_t::dkim, dkim_chunk_d, dkim_engine, dkim_eom_d, dkim_free_d, dkim_geterror_d, dkim_getresultstr_d, dkim_getsighdrx_d, DKIM_PROCESS_ALL, dkim_sign_d, magma_t::domain, magma_t::enabled, HEAP, magma_t::key, log_pedantic, magma, ns_empty(), ns_populated, NULLER_T, magma_t::selector, st_alloc_opts(), st_cleanup, st_data_get(), st_empty, st_length_get(), st_merge, st_populated, st_uchar_get(), stats_adjust_by_name(), and status.

Referenced by mail_add_forward_headers(), mail_add_outbound_headers(), smtp_bounce(), and smtp_reply().

5.234.3.4 int_t dkim_signature_verify (stringer_t * *id*, stringer_t * *message*)

[dkim.c](#) [dkim.c](#)

Note:

This function also updates the provider.dkim.* statistics.

Parameters:

id a managed string containing a printable string id for this message.

message a managed string containing the mail message data.

Returns:

1 if the dkim verification was successful, or < 0 otherwise. -1: General internal or dkim-related failure occurred, or no signature was present. -2: The signature was bad or the signing key was revoked or key retrieval failed (try again later).

Definition at line 221 of file dkim.c.

References dkim_chunk_d, dkim_engine, dkim_eom_d, dkim_free_d, dkim_getresultstr_d, dkim_verify_d, log_pedantic, st_data_get(), st_length_get(), stats_adjust_by_name(), and status.

Referenced by smtp_accept_message().

5.234.3.5 bool_t dkim_start (void)

Start the dkim engine.

Returns:

false on failure or true on success.

Definition at line 81 of file dkim.c.

References magma_t::dkim, dkim_engine, dkim_init_d, dkim_memory_alloc(), dkim_memory_free(), magma_t::enabled, file_load(), file_world_accessible(), magma_t::key, log_critical, log_pedantic, magma, ssl_verify_privkey(), st_char_get(), st_free(), and st_length_int().

Referenced by process_start().

5.234.3.6 void dkim_stop (void)

Stop the dkim engine.

Returns:

This function returns no value.

Definition at line 120 of file dkim.c.

References dkim_close_d, and dkim_engine.

Referenced by process_stop().

5.234.3.7 int_t dspam_check (uint64_t usernum, stringer_t * message, stringer_t ** signature)

[dspam.c](#) HIGH: Return a result structure with the disposition, confidence, probability and signature data. Then store the analysis result with the message. Either in the DB or by adding a custom header to the message. Return codes should become 0 for success, or a negative integer for errors. Add statistical updates to check and train functions. Result: result="\%s\"; class="\%s\"; probability=01.4f; confidence=02.2f; signature=lu; key=lu; Layman's terms: How spammy is this message

Definition at line 54 of file dspam.c.

References dspam_attach_d, dspam_create_d, dspam_destroy_d, dspam_detach_d, dspam_process_d, log_error, log_info, log_pedantic, MAGMA_SPOOL_DATA, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), spool_path(), sql_ping(), sql_pool, st_char_get(), st_free(), st_import(), and stmt_rebuild().

Referenced by smtp_accept_message().

5.234.3.8 bool_t dspam_start (void)

Definition at line 38 of file dspam.c.

References dspam_init_driver_d.

Referenced by process_start().

5.234.3.9 void dspam_stop (void)

Definition at line 45 of file dspam.c.

References dspam_shutdown_driver_d.

Referenced by process_stop().

5.234.3.10 bool_t dspam_train (uint64_t usernum, int_t disposition, stringer_t * signature)

Note:

The disposition parameter marks whether or not the signature is currently marked as junk. So training the signature means that it will toggle its status.

Parameters:

usenum the numerical id of the user making the spam training request.

disposition if 0, dspam will mark the signature as junk; otherwise, mark as OK.

signature a managed string containing the spam signature to be trained.

Returns:

true on success or false on failure.

Definition at line 173 of file dspam.c.

References dspam_attach_d, dspam_create_d, dspam_destroy_d, dspam_detach_d, dspam_process_d, log_info, log_pedantic, MAGMA_SPOOL_DATA, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), spool_path(), sql_ping(), sql_pool, st_char_get(), st_free(), st_length_get(), and stmt_rebuild().

Referenced by teacher_process().

5.234.3.11 bool_t lib_load_clamav (void)

[clamav.c](#) [clamav.c](#)

Returns:

false on failure or true on success.

Definition at line 459 of file clamav.c.

References lib_symbols(), M_BIND, and symbol_t.

Referenced by lib_load().

5.234.3.12 bool_t lib_load_dkim (void)

Initialize the dkim library and bind dynamically to the exported functions that are required.

Returns:

true on success or false on failure.

Definition at line 27 of file dkim.c.

References dkim_getsighdrx_d, dkim_libversion_d, dkim_version, lib_symbols(), M_BIND, and symbol_t.

Referenced by lib_load().

5.234.3.13 bool_t lib_load_dspam (void)

Initialize the dspam library and bind dynamically to the exported functions that are required.

Returns:

true on success or false on failure.

Definition at line 24 of file dspam.c.

References lib_symbols(), M_BIND, and symbol_t.

Referenced by lib_load().

5.234.3.14 bool_t lib_load_spf (void)

[spf.c](#) [spf.c](#)

Returns:

false on failure or true on success.

Definition at line 27 of file spf.c.

References lib_symbols(), M_BIND, SPF_get_lib_version_d, spf_version, and symbol_t.

Referenced by lib_load().

5.234.3.15 const char* lib_version_clamav (void)

Returns the version of ClamAV that was loaded at runtime.

Returns:

The ClamAV version as a constant string.

Definition at line 451 of file clamav.c.

References cl_retver_d.

Referenced by lib_load().

5.234.3.16 const chr_t* lib_version_dkim (void)

Return the version string of the dkim library.

Returns:

a pointer to a character string containing the dkim library version information.

Definition at line 19 of file dkim.c.

References dkim_version.

Referenced by lib_load().

5.234.3.17 chr_t* lib_version_dspam (void)

Return the version string of the dspam library.

Returns:

a pointer to a character string containing the dspam library version information.

Definition at line 16 of file dspam.c.

References dspam_version_d.

Referenced by lib_load().

5.234.3.18 const chr_t* lib_version_spf (void)

Return the version string of the spf library.

Returns:

a pointer to a character string containing the spf library version information.

Definition at line 19 of file spf.c.

References spf_version.

Referenced by lib_load().

5.234.3.19 int_t spf_check (void *ip, stringer_t *helo, stringer_t *mailfrom)

Validate an smtp request via spf.

Parameters:

ip an ip address object containing the ip address of the connection to be checked.

helo a managed string containing the client supplied HELO request value.

mailfrom a managed string containing the client supplied MAIL FROM request value. TODO: We really need to change these return values to account for 0

Returns:

-2 on spf check fail, -1 on other failure, and 1 on spf pass.

Definition at line 112 of file spf.c.

References ip_t::family, ip_t::ip4, ip_t::ip6, log_pedantic, mail_domain_get(), PL_RESERVED, placer_t, pool_get_obj(), pool_pull(), pool_release(), spf_pool, SPF_request_free_d, SPF_request_new_d, SPF_request_query_mailfrom_d, SPF_request_set_env_from_d, SPF_request_set_helo_dom_d, SPF_request_set_ipv4_d, SPF_request_set_ipv6_d, SPF_response_free_d, SPF_response_reason_d, SPF_response_result_d, SPF_sterror_d, SPF_streason_d, SPF_stresult_d, st_char_get(), st_empty, st_length_int(), and stats_adjust_by_name().

Referenced by smtp_rcpt_to().

5.234.3.20 bool_t spf_start (void)

Initialize the pool of spf server connections.

Returns:

false on failure or true on success.

Definition at line 52 of file spf.c.

References magma_t::iface, log_pedantic, magma, pool_alloc(), pool_set_obj(), magma_t::spf, spf_pool, and SPF_server_new_d.

Referenced by process_start().

5.234.3.21 void spf_stop (void)

Destroy the spf connection pool.

Returns:

This function returns no value.

Definition at line 84 of file spf.c.

References magma_t::iface, magma, pool_free(), pool_get_obj(), magma_t::spf, spf_pool, and SPF_server_free_d.

Referenced by process_stop().

5.234.3.22 int virus_check (stringer_t * data)

Virus scan a block of data.

Parameters:

data a managed string containing the block of data to be scanned.

Returns:

1 if the message passed the scan, or < 0 on failure. -1: general failure, or the virus scanner was not enabled. -2: the data matches a worm, trojan, or virus. -3: the data matches a phishing attempt.

Definition at line 361 of file clamav.c.

References cl_scandesc_d, cl_strerror_d, CONSTANT, magma_t::iface, log_error, log_pedantic, magma, MAGMA_SPOOL_SCAN, ns_length_get(), PLACER, spool_mktemp(), st_cmp_ci_starts(), st_data_get(), st_length_get(), stats_increment_by_name(), magma_t::virus, virus_engine, and virus_lock.

Referenced by portal_outbound_checks(), smtp_accept_message(), and smtp_data_outbound().

5.234.3.23 struct cl_engine* virus_engine_create (uint64_t * signatures) [read]

Generates a new ClamAV engine context.

Parameters:

signatures An optional pointer which will be used to record the number of signatures loaded.

Returns:

Returns a pointer to the newly created context or NULL if an error occurs.

Definition at line 86 of file clamav.c.

References `cl_engine_compile_d`, `cl_engine_free_d`, `cl_engine_new_d`, `cl_engine_set_num_d`, `cl_engine_set_str_d`, `cl_load_d`, `cl_strerror_d`, `magma_t::iface`, `log_error`, `magma`, `st_char_get()`, `magma_t::virus`, and `virus_spool`.

Referenced by `virus_engine_refresh()`, and `virus_start()`.

5.234.3.24 void virus_engine_destroy (struct cl_engine ** *target*)

Destroy a ClamAV engine context.

Parameters:

target the address of a pointer to a ClamAV engine context to be freed and reset.

Returns:

This function returns no value.

Definition at line 73 of file clamav.c.

References `cl_engine_free_d`, and `log_check`.

Referenced by `virus_engine_refresh()`, and `virus_stop()`.

5.234.3.25 int virus_engine_refresh (void)

Checks the virus database directory for new signatures. If new signatures are detected an updated ClamAV engine context is created.

Returns:

Returns 1 if the engine context is updated, 0 if no updates are necessary and -1 in the event of an error.

Definition at line 291 of file clamav.c.

References `cl_statchkdir_d`, `cl_statfree_d`, `cl_statinidir_d`, `magma_t::iface`, `log_error`, `log_info`, `magma`, `mm_wipe()`, `stats_increment_by_name()`, `stats_set_by_name()`, `magma_t::virus`, `virus_engine`, `virus_engine_create()`, `virus_engine_destroy()`, `virus_lock`, `virus_sigs`, `virus_sigs_total()`, and `virus_stat`.

Referenced by `process_maint()`.

5.234.3.26 uint64_t virus_sigs_loaded (void)

Get the number of virus signatures loaded by the ClamAV engine context.

Returns:

the number of virus signatures loaded by the ClamAV engine context.

Definition at line 39 of file clamav.c.

References `virus_lock`, and `virus_sigs`.

5.234.3.27 uint64_t virus_sigs_total (void)

Get the number of official signatures available inside the ClamAV signature directory.

See also:

`magma.iface.virus.signatures`

Returns:

the number of official signatures available inside the ClamAV signature directory, or 0 on failure.

Definition at line 55 of file clamav.c.

References `cl_countsigs_d`, `cl_strerror_d`, `magma_t::iface`, `log_error`, `magma`, and `magma_t::virus`.

Referenced by `virus_engine_refresh()`, and `virus_start()`.

5.234.3.28 bool_t virus_start (void)

Initializes the global ClamAV engine context and configures it appropriately.

Returns:

Returns true if the ClamAV engine was loaded correctly.

Definition at line 196 of file clamav.c.

References `cl_init_d`, `cl_statfree_d`, `cl_statinidir_d`, `cl_strerror_d`, `magma_t::iface`, `log_critical`, `log_pedantic`, `magma`, `MAGMA_SPOOL_SCAN`, `mm_wipe()`, `magma_t::spool`, `spool_check()`, `spool_path()`, `st_char_get()`, `st_length_int()`, `stats_increment_by_name()`, `stats_set_by_name()`, `magma_t::virus`, `virus_engine`, `virus_engine_create()`, `virus_sigs`, `virus_sigs_total()`, `virus_spool`, and `virus_stat`.

Referenced by `process_start()`.

5.234.3.29 void virus_stop (void)

Shut down the ClamAV library and free all its data and temporary files.

Returns:

This function returns no value.

Definition at line 249 of file clamav.c.

References `cl_shutdown_d`, `cl_statfree_d`, `magma_t::iface`, `magma`, `st_free()`, `stats_set_by_name()`, `magma_t::virus`, `virus_engine`, `virus_engine_destroy()`, `virus_sigs`, `virus_spool`, and `virus_stat`.

Referenced by `process_stop()`.

5.235 src/providers/checkers/clamav.c File Reference

```
#include "magma.h"
```

Functions

- uint64_t [virus_sigs_loaded](#) (void)
Get the number of virus signatures loaded by the ClamAV engine context.
- uint64_t [virus_sigs_total](#) (void)
Get the number of official signatures available inside the ClamAV signature directory.
- void [virus_engine_destroy](#) (struct cl_engine **target)
Destroy a ClamAV engine context.
- struct cl_engine * [virus_engine_create](#) (uint64_t *signatures)
- bool_t [virus_start](#) (void)
- void [virus_stop](#) (void)
Shut down the ClamAV library and free all its data and temporary files.
- int [virus_engine_refresh](#) (void)
- int [virus_check](#) (stringer_t *data)
Virus scan a block of data.
- const char * [lib_version_clamav](#) (void)
- bool_t [lib_load_clamav](#) (void)
Loads the external functions needed by the ClamAV interface.

Variables

- stringer_t * [virus_spool](#) = NULL
- struct cl_stat [virus_stat](#)
- unsigned int [virus_sigs](#) = 0
- struct cl_engine * [virus_engine](#) = NULL
- pthread_rwlock_t [virus_lock](#) = PTHREAD_RWLOCK_INITIALIZER

5.235.1 Function Documentation

5.235.1.1 bool_t lib_load_clamav (void)

Loads the external functions needed by the ClamAV interface. [clamav.c](#)

Returns:

false on failure or true on success.

Definition at line 459 of file clamav.c.

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.235.1.2 const char* lib_version_clamav (void)

Returns the version of ClamAV that was loaded at runtime.

Returns:

The ClamAV version as a constant string.

Definition at line 451 of file clamav.c.

References `cl_retver_d`.

Referenced by `lib_load()`.

5.235.1.3 int virus_check (stringer_t * data)

Virus scan a block of data.

Parameters:

data a managed string containing the block of data to be scanned.

Returns:

1 if the message passed the scan, or < 0 on failure. -1: general failure, or the virus scanner was not enabled. -2: the data matches a worm, trojan, or virus. -3: the data matches a phishing attempt.

Definition at line 361 of file clamav.c.

References `cl_scandesc_d`, `cl_strerror_d`, `CONSTANT`, `magma_t::iface`, `log_error`, `log_pedantic`, `magma`, `MAGMA_SPOOL_SCAN`, `ns_length_get()`, `PLACER`, `spool_mktemp()`, `st_cmp_ci_starts()`, `st_data_get()`, `st_length_get()`, `stats_increment_by_name()`, `magma_t::virus`, `virus_engine`, and `virus_lock`.

Referenced by `portal_outbound_checks()`, `smtp_accept_message()`, and `smtp_data_outbound()`.

5.235.1.4 struct cl_engine* virus_engine_create (uint64_t * signatures) [read]

Generates a new ClamAV engine context.

Parameters:

signatures An optional pointer which will be used to record the number of signatures loaded.

Returns:

Returns a pointer to the newly created context or NULL if an error occurs.

Definition at line 86 of file clamav.c.

References `cl_engine_compile_d`, `cl_engine_free_d`, `cl_engine_new_d`, `cl_engine_set_num_d`, `cl_engine_set_str_d`, `cl_load_d`, `cl_strerror_d`, `magma_t::iface`, `log_error`, `magma`, `st_char_get()`, `magma_t::virus`, and `virus_spool`.

Referenced by `virus_engine_refresh()`, and `virus_start()`.

5.235.1.5 void virus_engine_destroy (struct cl_engine ** target)

Destroy a ClamAV engine context.

Parameters:

target the address of a pointer to a ClamAV engine context to be freed and reset.

Returns:

This function returns no value.

Definition at line 73 of file clamav.c.

References `cl_engine_free_d`, and `log_check`.

Referenced by `virus_engine_refresh()`, and `virus_stop()`.

5.235.1.6 int virus_engine_refresh (void)

Checks the virus database directory for new signatures. If new signatures are detected an updated ClamAV engine context is created.

Returns:

Returns 1 if the engine context is updated, 0 if no updates are necessary and -1 in the event of an error.

Definition at line 291 of file clamav.c.

References `cl_statchkdir_d`, `cl_statfree_d`, `cl_statinidir_d`, `magma_t::iface`, `log_error`, `log_info`, `magma`, `mm_wipe()`, `stats_increment_by_name()`, `stats_set_by_name()`, `magma_t::virus`, `virus_engine`, `virus_engine_create()`, `virus_engine_destroy()`, `virus_lock`, `virus_sigs`, `virus_sigs_total()`, and `virus_stat`.

Referenced by `process_maint()`.

5.235.1.7 uint64_t virus_sigs_loaded (void)

Get the number of virus signatures loaded by the ClamAV engine context.

Returns:

the number of virus signatures loaded by the ClamAV engine context.

Definition at line 39 of file clamav.c.

References `virus_lock`, and `virus_sigs`.

5.235.1.8 uint64_t virus_sigs_total (void)

Get the number of official signatures available inside the ClamAV signature directory.

See also:

`magma.iface.virus.signatures`

Returns:

the number of official signatures available inside the ClamAV signature directory, or 0 on failure.

Definition at line 55 of file clamav.c.

References `cl_countsigs_d`, `cl_strerror_d`, `magma_t::iface`, `log_error`, `magma`, and `magma_t::virus`.

Referenced by `virus_engine_refresh()`, and `virus_start()`.

5.235.1.9 bool_t virus_start (void)

Initializes the global ClamAV engine context and configures it appropriately.

Returns:

Returns true if the ClamAV engine was loaded correctly.

Definition at line 196 of file clamav.c.

References `cl_init_d`, `cl_statfree_d`, `cl_statinidir_d`, `cl_strerror_d`, `magma_t::iface`, `log_critical`, `log_pedantic`, `magma`, `MAGMA_SPOOL_SCAN`, `mm_wipe()`, `magma_t::spool`, `spool_check()`, `spool_path()`, `st_char_get()`, `st_length_int()`, `stats_increment_by_name()`, `stats_set_by_name()`, `magma_t::virus`, `virus_engine`, `virus_engine_create()`, `virus_sigs`, `virus_sigs_total()`, `virus_spool`, and `virus_stat`.

Referenced by `process_start()`.

5.235.1.10 void virus_stop (void)

Shut down the ClamAV library and free all its data and temporary files.

Returns:

This function returns no value.

Definition at line 249 of file clamav.c.

References `cl_shutdown_d`, `cl_statfree_d`, `magma_t::iface`, `magma`, `st_free()`, `stats_set_by_name()`, `magma_t::virus`, `virus_engine`, `virus_engine_destroy()`, `virus_sigs`, `virus_spool`, and `virus_stat`.

Referenced by `process_stop()`.

5.235.2 Variable Documentation**5.235.2.1 struct cl_engine* virus_engine = NULL**

The virus engine context pointer.

Definition at line 28 of file clamav.c.

Referenced by `virus_check()`, `virus_engine_refresh()`, `virus_start()`, and `virus_stop()`.

5.235.2.2 pthread_rwlock_t virus_lock = PTHREAD_RWLOCK_INITIALIZER

The virus engine read/write lock.

Definition at line 33 of file clamav.c.

Referenced by `virus_check()`, `virus_engine_refresh()`, and `virus_sigs_loaded()`.

5.235.2.3 unsigned int virus_sigs = 0

The number of signatures loaded by the virus engine.

Definition at line 23 of file clamav.c.

Referenced by `virus_engine_refresh()`, `virus_sigs_loaded()`, `virus_start()`, and `virus_stop()`.

5.235.2.4 stringer_t* virus_spool = NULL

The virus engine spool directory.

Definition at line 13 of file clamav.c.

Referenced by `virus_engine_create()`, `virus_start()`, and `virus_stop()`.

5.235.2.5 struct cl_stat virus_stat

The status of the signatures directory.

Definition at line 18 of file clamav.c.

Referenced by virus_engine_refresh(), virus_start(), and virus_stop().

5.236 src/providers/checkers/dkim.c File Reference

```
#include "magma.h"
```

Defines

- `#define DKIM_PROCESS_ALL -1L`

Functions

- `const chr_t * lib_version_dkim (void)`
Return the version string of the dkim library.
- `bool_t lib_load_dkim (void)`
Initialize the dkim library and bind dynamically to the exported functions that are required.
- `void * dkim_memory_alloc (void *closure, size_t nbytes)`
A memory allocation routine stub for dkim.
- `void dkim_memory_free (void *closure, void *ptr)`
A memory free routine stub for dkim.
- `bool_t dkim_start (void)`
Start the dkim engine.
- `void dkim_stop (void)`
Stop the dkim engine.
- `stringer_t * dkim_signature_create (stringer_t *id, stringer_t *domain, stringer_t *message)`
Generate a DKIM signature for a message.
- `int_t dkim_signature_verify (stringer_t *id, stringer_t *message)`
Perform dkim verification of a signed message.

Variables

- `chr_t dkim_version [8]`
- `DKIM_LIB * dkim_engine = NULL`

5.236.1 Define Documentation

5.236.1.1 #define DKIM_PROCESS_ALL -1L

Definition at line 13 of file dkim.c.

Referenced by dkim_signature_create().

5.236.2 Function Documentation

5.236.2.1 void* dkim_memory_alloc (void * *closure*, size_t *nbytes*)

A memory allocation routine stub for dkim.

Note:

This function is passed to dkim_init().

Parameters:

closure a void pointer to a memory closure passed to dkim_sign() and dkim_verify().

nbytes the size, in bytes, of the memory block to be allocated.

Returns:

a pointer to a newly allocated block of memory of specified size.

Definition at line 59 of file dkim.c.

References mm_alloc().

Referenced by dkim_start().

5.236.2.2 void dkim_memory_free (void * *closure*, void * *ptr*)

A memory free routine stub for dkim.

Note:

This function is passed to dkim_init().

Parameters:

closure a void pointer to a memory closure passed to dkim_sign() and dkim_verify().

ptr a pointer to the block of memory to be released.

Returns:

This function returns no value.

Definition at line 71 of file dkim.c.

References mm_free().

Referenced by dkim_start().

5.236.2.3 stringer_t* dkim_signature_create (stringer_t * *id*, stringer_t * *domain*, stringer_t * *message*)

Generate a DKIM signature for a message.

Note:

For this function to work, the [magma.dkim.enabled](#) configuration option must be set. This function also updates the provider.dkim.signed statistic.

Parameters:

id a managed string containing a printable string id for this message.

domain the domain name hosting the identified DKIM public key.

message a managed string containing the mail message data.

Returns:

NULL on failure or if the global [magma.dkim.enabled](#) setting is false; otherwise, a managed string containing a DKIM-Signature header built using the dkim signature that was generated for the input message.

Definition at line 141 of file dkim.c.

References CONTIGUOUS, magma_t::dkim, dkim_chunk_d, dkim_engine, dkim_eom_d, dkim_free_d, dkim_geterror_d, dkim_getresultstr_d, dkim_getsighdrx_d, DKIM_PROCESS_ALL, dkim_sign_d, magma_t::domain, magma_t::enabled, HEAP, magma_t::key, log_pedantic, magma, ns_empty(), ns_populated, NULLER_T, magma_t::selector, st_alloc_opts(), st_cleanup, st_data_get(), st_empty, st_length_get(), st_merge, st_populated, st_uchar_get(), stats_adjust_by_name(), and status.

Referenced by mail_add_forward_headers(), mail_add_outbound_headers(), smtp_bounce(), and smtp_reply().

5.236.2.4 int_t dkim_signature_verify (stringer_t * id, stringer_t * message)

Perform dkim verification of a signed message. [dkim.c](#)

Note:

This function also updates the provider.dkim.* statistics.

Parameters:

id a managed string containing a printable string id for this message.

message a managed string containing the mail message data.

Returns:

1 if the dkim verification was successful, or < 0 otherwise. -1: General internal or dkim-related failure occurred, or no signature was present. -2: The signature was bad or the signing key was revoked or key retrieval failed (try again later).

Definition at line 221 of file dkim.c.

References dkim_chunk_d, dkim_engine, dkim_eom_d, dkim_free_d, dkim_getresultstr_d, dkim_verify_d, log_pedantic, st_data_get(), st_length_get(), stats_adjust_by_name(), and status.

Referenced by smtp_accept_message().

5.236.2.5 bool_t dkim_start (void)

Start the dkim engine.

Returns:

false on failure or true on success.

Definition at line 81 of file dkim.c.

References magma_t::dkim, dkim_engine, dkim_init_d, dkim_memory_alloc(), dkim_memory_free(), magma_t::enabled, file_load(), file_world_accessible(), magma_t::key, log_critical, log_pedantic, magma, ssl_verify_privkey(), st_char_get(), st_free(), and st_length_int().

Referenced by process_start().

5.236.2.6 void dkim_stop (void)

Stop the dkim engine.

Returns:

This function returns no value.

Definition at line 120 of file dkim.c.

References dkim_close_d, and dkim_engine.

Referenced by process_stop().

5.236.2.7 bool_t lib_load_dkim (void)

Initialize the dkim library and bind dynamically to the exported functions that are required.

Returns:

true on success or false on failure.

Definition at line 27 of file dkim.c.

References dkim_getsighdrx_d, dkim_libversion_d, dkim_version, lib_symbols(), M_BIND, and symbol_t.

Referenced by lib_load().

5.236.2.8 const chr_t* lib_version_dkim (void)

Return the version string of the dkim library.

Returns:

a pointer to a character string containing the dkim library version information.

Definition at line 19 of file dkim.c.

References dkim_version.

Referenced by lib_load().

5.236.3 Variable Documentation

5.236.3.1 DKIM_LIB* dkim_engine = NULL

Definition at line 11 of file dkim.c.

Referenced by dkim_signature_create(), dkim_signature_verify(), dkim_start(), and dkim_stop().

5.236.3.2 chr_t dkim_version[8]

Definition at line 10 of file dkim.c.

Referenced by lib_load_dkim(), and lib_version_dkim().

5.237 src/providers/checkers/dspam.c File Reference

```
#include "magma.h"
```

Functions

- [chr_t * lib_version_dspam](#) (void)
Return the version string of the dspam library.
- [bool_t lib_load_dspam](#) (void)
Initialize the dspam library and bind dynamically to the exported functions that are required.
- [bool_t dspam_start](#) (void)
- [void dspam_stop](#) (void)
- [int_t dspam_check](#) (uint64_t usernum, [stringer_t](#) *message, [stringer_t](#) **signature)
dspam.c
- [bool_t dspam_train](#) (uint64_t usernum, [int_t](#) disposition, [stringer_t](#) *signature)

Variables

- [pool_t * sql_pool](#)

5.237.1 Function Documentation

5.237.1.1 [int_t dspam_check](#) (uint64_t usernum, [stringer_t](#) * message, [stringer_t](#) ** signature)

[dspam.c](#) HIGH: Return a result structure with the disposition, confidence, probability and signature data. Then store the analysis result with the message. Either in the DB or by adding a custom header to the message. Return codes should become 0 for success, or a negative integer for errors. Add statistical updates to check and train functions. Result: result="\%s\"; class="\%s\"; probability=01.4f; confidence=02.2f; signature=lu; key=lu; Layman's terms: How spammy is this message

Definition at line 54 of file dspam.c.

References [dspam_attach_d](#), [dspam_create_d](#), [dspam_destroy_d](#), [dspam_detach_d](#), [dspam_process_d](#), [log_error](#), [log_info](#), [log_pedantic](#), [MAGMA_SPOOL_DATA](#), [PL_RESERVED](#), [pool_get_obj\(\)](#), [pool_pull\(\)](#), [pool_release\(\)](#), [spool_path\(\)](#), [sql_ping\(\)](#), [sql_pool](#), [st_char_get\(\)](#), [st_free\(\)](#), [st_import\(\)](#), and [stmt_rebuild\(\)](#).

Referenced by [smtp_accept_message\(\)](#).

5.237.1.2 [bool_t dspam_start](#) (void)

Definition at line 38 of file dspam.c.

References [dspam_init_driver_d](#).

Referenced by [process_start\(\)](#).

5.237.1.3 [void dspam_stop](#) (void)

Definition at line 45 of file dspam.c.

References [dspam_shutdown_driver_d](#).

Referenced by [process_stop\(\)](#).

5.237.1.4 bool_t dspam_train (uint64_t usernum, int_t disposition, stringer_t * signature)**Note:**

The disposition parameter marks whether or not the signature is currently marked as junk. So training the signature means that it will toggle its status.

Parameters:

usernum the numerical id of the user making the spam training request.

disposition if 0, dspam will mark the signature as junk; otherwise, mark as OK.

signature a managed string containing the spam signature to be trained.

Returns:

true on success or false on failure.

Definition at line 173 of file dspam.c.

References dspam_attach_d, dspam_create_d, dspam_destroy_d, dspam_detach_d, dspam_process_d, log_info, log_pedantic, MAGMA_SPOOL_DATA, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), spool_path(), sql_ping(), sql_pool, st_char_get(), st_free(), st_length_get(), and stmt_rebuild().

Referenced by teacher_process().

5.237.1.5 bool_t lib_load_dspam (void)

Initialize the dspam library and bind dynamically to the exported functions that are required.

Returns:

true on success or false on failure.

Definition at line 24 of file dspam.c.

References lib_symbols(), M_BIND, and symbol_t.

Referenced by lib_load().

5.237.1.6 chr_t* lib_version_dspam (void)

Return the version string of the dspam library.

Returns:

a pointer to a character string containing the dspam library version information.

Definition at line 16 of file dspam.c.

References dspam_version_d.

Referenced by lib_load().

5.237.2 Variable Documentation**5.237.2.1 pool_t* sql_pool**

Definition at line 143 of file mysql.c.

Referenced by `dspam_check()`, `dspam_train()`, `mail_db_update_message_folder()`, `sql_insert()`, `sql_insert_conn()`, `sql_num_rows()`, `sql_num_rows_conn()`, `sql_ping()`, `sql_query()`, `sql_query_conn()`, `sql_query_res()`, `sql_query_res_conn()`, `sql_start()`, `sql_stop()`, `sql_write()`, `sql_write_conn()`, `stmt_exec()`, `stmt_exec_affected()`, `stmt_get_result()`, `stmt_insert()`, `stmt_rebuild()`, `stmt_start()`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

5.238 src/providers/checkers/spf.c File Reference

```
#include "magma.h"
```

Functions

- `const chr_t * lib_version_spf` (void)
Return the version string of the spf library.
- `bool_t lib_load_spf` (void)
Initialize the spf library and bind dynamically to the exported functions that are required.
- `bool_t spf_start` (void)
Initialize the pool of spf server connections.
- `void spf_stop` (void)
Destroy the spf connection pool.
- `int_t spf_check` (void *ip, `stringer_t` *helo, `stringer_t` *mailfrom)
Validate an smtp request via spf.

Variables

- `pool_t * spf_pool` = NULL
- `chr_t spf_version` [8]

5.238.1 Function Documentation

5.238.1.1 `bool_t lib_load_spf` (void)

Initialize the spf library and bind dynamically to the exported functions that are required. [spf.c](#)

Returns:

false on failure or true on success.

Definition at line 27 of file `spf.c`.

References `lib_symbols()`, `M_BIND`, `SPF_get_lib_version_d`, `spf_version`, and `symbol_t`.

Referenced by `lib_load()`.

5.238.1.2 `const chr_t* lib_version_spf` (void)

Return the version string of the spf library.

Returns:

a pointer to a character string containing the spf library version information.

Definition at line 19 of file `spf.c`.

References `spf_version`.

Referenced by `lib_load()`.

5.238.1.3 `int_t spf_check (void * ip, stringer_t * helo, stringer_t * mailfrom)`

Validate an smtp request via spf.

Parameters:

ip an ip address object containing the ip address of the connection to be checked.

helo a managed string containing the client supplied HELO request value.

mailfrom a managed string containing the client supplied MAIL FROM request value. TODO: We really need to change these return values to account for 0

Returns:

-2 on spf check fail, -1 on other failure, and 1 on spf pass.

Definition at line 112 of file spf.c.

References `ip_t::family`, `ip_t::ip4`, `ip_t::ip6`, `log_pedantic`, `mail_domain_get()`, `PL_RESERVED`, `placer_t`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `spf_pool`, `SPF_request_free_d`, `SPF_request_new_d`, `SPF_request_query_mailfrom_d`, `SPF_request_set_env_from_d`, `SPF_request_set_helo_dom_d`, `SPF_request_set_ipv4_d`, `SPF_request_set_ipv6_d`, `SPF_response_free_d`, `SPF_response_reason_d`, `SPF_response_result_d`, `SPF_strerror_d`, `SPF_strerror_d`, `SPF_strerror_d`, `st_char_get()`, `st_empty`, `st_length_int()`, and `stats_adjust_by_name()`.

Referenced by `smtp_rcpt_to()`.

5.238.1.4 `bool_t spf_start (void)`

Initialize the pool of spf server connections.

Returns:

false on failure or true on success.

Definition at line 52 of file spf.c.

References `magma_t::iface`, `log_pedantic`, `magma`, `pool_alloc()`, `pool_set_obj()`, `magma_t::spf`, `spf_pool`, and `SPF_server_new_d`.

Referenced by `process_start()`.

5.238.1.5 `void spf_stop (void)`

Destroy the spf connection pool.

Returns:

This function returns no value.

Definition at line 84 of file spf.c.

References `magma_t::iface`, `magma`, `pool_free()`, `pool_get_obj()`, `magma_t::spf`, `spf_pool`, and `SPF_server_free_d`.

Referenced by `process_stop()`.

5.238.2 Variable Documentation

5.238.2.1 `pool_t* spf_pool = NULL`

Definition at line 12 of file spf.c.

Referenced by `spf_check()`, `spf_start()`, and `spf_stop()`.

5.238.2.2 chr_t spf_version[8]

Definition at line 13 of file spf.c.

Referenced by lib_load_spf(), and lib_version_spf().

5.239 src/providers/compress/bzip.c File Reference

```
#include "magma.h"
```

Functions

- `const char * lib_version_bzip` (void)
Return the version string of the bzip library.
- `bool_t lib_load_bzip` (void)
Initialize the bzip library and bind dynamically to the exported functions that are required.
- `stringer_t * decompress_bzip` (`compress_t *compressed`)
Decompress data using the bzip engine.
- `compress_t * compress_bzip` (`stringer_t *input`)
Compress data using the bzip engine.

Variables

- `char bzip_version` [16]

5.239.1 Function Documentation

5.239.1.1 `compress_t* compress_bzip` (`stringer_t * input`)

Compress data using the bzip engine.

Parameters:

input a managed string containing the data to be compressed.

Returns:

NULL on failure, or a pointer to the head of the compressed data on success.

Definition at line 100 of file bzip.c.

References `BZ2_bzBuffToBuffCompress_d`, `compress_alloc()`, `compress_body_data()`, `COMPRESS_ENGINE_BZIP`, `compress_free()`, `compress_head_t`, `decompress_bzip()`, `hash_adler32()`, `log_info`, `st_data_get()`, `st_free()`, and `st_length_get()`.

Referenced by `engine_compress()`, and `tank_store()`.

5.239.1.2 `stringer_t* decompress_bzip` (`compress_t * compressed`)

Decompress data using the bzip engine.

Parameters:

compressed a pointer to the head of the compressed data.

Returns:

NULL on failure, or a managed string containing the uncompressed data on success.

Definition at line 61 of file bzip.c.

References BZ2_bzBuffToBuffDecompress_d, compress_body_data(), COMPRESS_ENGINE_BZIP, compress_head_t, hash_adler32(), log_info, st_alloc(), st_data_get(), st_free(), and st_length_set().

Referenced by compress_bzip(), engine_decompress(), and tank_load().

5.239.1.3 bool_t lib_load_bzip (void)

Initialize the bzip library and bind dynamically to the exported functions that are required. [bzip.c](#)

Returns:

true on success or false on failure.

Definition at line 24 of file bzip.c.

References BZ2_bzlibVersion_d, bzip_version, lib_symbols(), log_pedantic, M_BIND, ns_length_int(), and symbol_t.

Referenced by lib_load().

5.239.1.4 const char* lib_version_bzip (void)

Return the version string of the bzip library.

Returns:

a pointer to a character string containing the bzip library version information.

Definition at line 16 of file bzip.c.

References bzip_version.

Referenced by lib_load().

5.239.2 Variable Documentation

5.239.2.1 char bzip_version[16]

Definition at line 10 of file bzip.c.

Referenced by lib_load_bzip(), and lib_version_bzip().

5.240 src/providers/compress/compress.c File Reference

```
#include "magma.h"
```

Functions

- `size_t compress_block_length` (void)
Get the compression engine's (LZOIX-1) block size.
- `uint64_t compress_total_length` (compress_t *buffer)
Return the total length of a compressed header and body.
- `uint64_t compress_orig_hash` (compress_t *buffer)
Get the hash value of a compressed buffer's original data.
- `uint64_t compress_orig_length` (compress_t *buffer)
Get the original length of a compressed buffer's data.
- `uint64_t compress_body_hash` (compress_t *buffer)
Get the hash value of a compressed buffer's (compressed) body.
- `uint64_t compress_body_length` (compress_t *buffer)
Return the length of a compressed buffer's body.
- `void * compress_body_data` (compress_t *buffer)
Return the compressed body of data associated with a compressed header.
- `size_t compress_body_offset` (void)
Get the offset to the compressed body from the compressed header.
- `compress_t * compress_import` (stringer_t *s)
Parse and validate a managed string with compressed data and return a compressed header.
- `compress_t * compress_alloc` (size_t length)
Allocate a new compressed object.
- `void compress_free` (compress_t *buffer)
Free a compressed object.
- `void compress_cleanup` (compress_t *buffer)
A checked cleanup function which can be used free a compressed object.

5.240.1 Function Documentation

5.240.1.1 compress_t* compress_alloc (size_t length)

Allocate a new compressed object. [compress.c](#)

Parameters:

length the size, in bytes, of the data after compression.

Returns:

a pointer to the head of the newly allocated compressed object.

Definition at line 140 of file compress.c.

References `compress_head_t`, and `mm_alloc()`.

Referenced by `compress_bzip()`, `compress_lzo()`, and `compress_zlib()`.

5.240.1.2 size_t compress_block_length (void)

Get the compression engine's (LZO1X-1) block size.

Note:

In the future this might get stored in the compression header and/or change depending on the engine being used.

Returns:

the size, in bytes, of the compression engine's block size.

Definition at line 15 of file compress.c.

5.240.1.3 void* compress_body_data (compress_t * *buffer*)

Return the compressed body of data associated with a compressed header.

Parameters:

buffer the input compressed header.

Returns:

a pointer to the body (start) of the compressed data.

Definition at line 81 of file compress.c.

References `compress_head_t`.

Referenced by `compress_bzip()`, `compress_import()`, `compress_lzo()`, `compress_zlib()`, `decompress_bzip()`, `decompress_lzo()`, and `decompress_zlib()`.

5.240.1.4 uint64_t compress_body_hash (compress_t * *buffer*)

Get the hash value of a compressed buffer's (compressed) body.

Parameters:

buffer a pointer to the header of the compressed data.

Returns:

the hash value of the (compressed) of the compressed data.

Definition at line 58 of file compress.c.

References `compress_head_t`.

5.240.1.5 `uint64_t compress_body_length (compress_t * buffer)`

Return the length of a compressed buffer's body.

Returns:

the total length in bytes of the compressed data body, excluding the compressed header.

Definition at line 69 of file `compress.c`.

References `compress_head_t`.

Referenced by `compress_total_length()`, and `tank_store()`.

5.240.1.6 `size_t compress_body_offset (void)`

Get the offset to the compressed body from the compressed header.

Parameters:

the size, in bytes, of the offset from the start of the compressed data body from the start of the compressed header.

Definition at line 90 of file `compress.c`.

References `compress_head_t`.

5.240.1.7 `void compress_cleanup (compress_t * buffer)`

A checked cleanup function which can be used free a compressed object.

Parameters:

buffer a pointer to the head of the compressed object to be freed.

Returns:

This function returns no value.

Definition at line 161 of file `compress.c`.

References `mm_free()`.

Referenced by `mail_store_message()`.

5.240.1.8 `void compress_free (compress_t * buffer)`

Free a compressed object.

Parameters:

buffer a pointer to the head of the compressed object to be freed.

Returns:

This function returns no value.

Definition at line 150 of file `compress.c`.

References `mm_free()`.

Referenced by `compress_bzip()`, `compress_lzo()`, `compress_zlib()`, and `tank_store()`.

5.240.1.9 compress_t* compress_import (stringer_t * s)

Parse and validate a managed string with compressed data and return a compressed header.

Parameters:

s the managed string containing the compressed data.

Returns:

NULL on general failure or if the Adler 32 hash doesn't match, or a pointer to the data's compressed header on success.

Definition at line 100 of file compress.c.

References compress_body_data(), compress_head_t, hash_adler32(), log_pedantic, st_data_get(), st_empty, and st_length_get().

Referenced by mail_load_message().

5.240.1.10 uint64_t compress_orig_hash (compress_t * *buffer*)

Get the hash value of a compressed buffer's original data.

Parameters:

buffer a pointer to the head of the compressed data.

Returns:

the hash value of the original (uncompressed) data.

Definition at line 34 of file compress.c.

References compress_head_t.

5.240.1.11 uint64_t compress_orig_length (compress_t * *buffer*)

Get the original length of a compressed buffer's data.

Parameters:

buffer a pointer to the head of the compressed data.

Returns:

the original length of the (uncompressed) data.

Definition at line 46 of file compress.c.

References compress_head_t.

5.240.1.12 uint64_t compress_total_length (compress_t * *buffer*)

Return the total length of a compressed header and body.

Returns:

the total length, in bytes, of the compressed header and body.

Definition at line 24 of file compress.c.

References compress_body_length(), and compress_head_t.

Referenced by mail_store_message(), and tank_store().

5.241 src/providers/compress/compress.h File Reference

Typedefs

- typedef [stringer_t](#) [compress_t](#)

Enumerations

- enum { [COMPRESS_ENGINE_LZO](#) = 1, [COMPRESS_ENGINE_ZLIB](#) = 2, [COMPRESS_ENGINE_BZIP](#) = 4 }

Functions

- struct [__attribute__](#) ((packed))
- [bool_t](#) [lib_load_bzip](#) (void)
bzip.c
- const char * [lib_version_bzip](#) (void)
Return the version string of the bzip library.
- [compress_t](#) * [compress_bzip](#) ([stringer_t](#) *input)
Compress data using the bzip engine.
- [stringer_t](#) * [decompress_bzip](#) ([compress_t](#) *compressed)
Decompress data using the bzip engine.
- [compress_t](#) * [compress_alloc](#) (size_t length)
compress.c
- size_t [compress_block_length](#) (void)
Get the compression engine's (LZO1X-1) block size.
- void * [compress_body_data](#) ([compress_t](#) *buffer)
Return the compressed body of data associated with a compressed header.
- uint64_t [compress_body_hash](#) ([compress_t](#) *buffer)
Get the hash value of a compressed buffer's (compressed) body.
- uint64_t [compress_body_length](#) ([compress_t](#) *buffer)
Return the length of a compressed buffer's body.
- size_t [compress_body_offset](#) (void)
Get the offset to the compressed body from the compressed header.
- void [compress_free](#) ([compress_t](#) *buffer)
Free a compressed object.
- [compress_t](#) * [compress_import](#) ([stringer_t](#) *s)
Parse and validate a managed string with compressed data and return a compressed header.
- uint64_t [compress_orig_hash](#) ([compress_t](#) *buffer)
Get the hash value of a compressed buffer's original data.

- `uint64_t compress_orig_length (compress_t *buffer)`
Get the original length of a compressed buffer's data.
- `uint64_t compress_total_length (compress_t *buffer)`
Return the total length of a compressed header and body.
- `void compress_cleanup (compress_t *buffer)`
A checked cleanup function which can be used free a compressed object.
- `compress_t * engine_compress (uint8_t engine, stringer_t *s)`
engine.c
- `stringer_t * engine_decompress (compress_t *buffer)`
- `bool_t lib_load_lzo (void)`
lzo.c
- `compress_t * compress_lzo (stringer_t *input)`
Compress data using the lzo engine.
- `const char * lib_version_lzo (void)`
Return the version string of the lzo library.
- `stringer_t * decompress_block_lzo (stringer_t *block)`
Decompress a single block of data using the lzo engine.
- `stringer_t * decompress_lzo (compress_t *compressed)`
Decompress data using the lzo engine.
- `bool_t lib_load_zlib (void)`
zlib.c
- `const char * lib_version_zlib (void)`
Return the version string of zlib.
- `compress_t * compress_zlib (stringer_t *input)`
Compress a block of data using the zlib engine.
- `stringer_t * decompress_zlib (compress_t *compressed)`
Decompress a block of data using the zlib engine.

Variables

- `enum { ... } COMPRESS_ENGINE`
- `compress_head_t`

5.241.1 Typedef Documentation

5.241.1.1 typedef stringer_t compress_t

Definition at line 33 of file `compress.h`.

5.241.2 Enumeration Type Documentation

5.241.2.1 anonymous enum

Enumerator:

COMPRESS_ENGINE_LZO

COMPRESS_ENGINE_ZLIB

COMPRESS_ENGINE_BZIP

Definition at line 11 of file compress.h.

5.241.3 Function Documentation

5.241.3.1 struct __attribute__((packed)) [read]

Definition at line 17 of file compress.h.

References length.

5.241.3.2 compress_t* compress_alloc (size_t length)

[compress.c](#) [compress.c](#)

Parameters:

length the size, in bytes, of the data after compression.

Returns:

a pointer to the head of the newly allocated compressed object.

Definition at line 140 of file compress.c.

References compress_head_t, and mm_alloc().

Referenced by compress_bzip(), compress_lzo(), and compress_zlib().

5.241.3.3 size_t compress_block_length (void)

Get the compression engine's (LZO1X-1) block size.

Note:

In the future this might get stored in the compression header and/or change depending on the engine being used.

Returns:

the size, in bytes, of the compression engine's block size.

Definition at line 15 of file compress.c.

5.241.3.4 void* compress_body_data (compress_t * buffer)

Return the compressed body of data associated with a compressed header.

Parameters:

buffer the input compressed header.

Returns:

a pointer to the body (start) of the compressed data.

Definition at line 81 of file compress.c.

References compress_head_t.

Referenced by compress_bzip(), compress_import(), compress_lzo(), compress_zlib(), decompress_bzip(), decompress_lzo(), and decompress_zlib().

5.241.3.5 uint64_t compress_body_hash (compress_t * *buffer*)

Get the hash value of a compressed buffer's (compressed) body.

Parameters:

buffer a pointer to the header of the compressed data.

Returns:

the hash value of the (compressed) of the compressed data.

Definition at line 58 of file compress.c.

References compress_head_t.

5.241.3.6 uint64_t compress_body_length (compress_t * *buffer*)

Return the length of a compressed buffer's body.

Returns:

the total length in bytes of the compressed data body, excluding the compressed header.

Definition at line 69 of file compress.c.

References compress_head_t.

Referenced by compress_total_length(), and tank_store().

5.241.3.7 size_t compress_body_offset (void)

Get the offset to the compressed body from the compressed header.

Parameters:

the size, in bytes, of the offset from the start of the compressed data body from the start of the compressed header.

Definition at line 90 of file compress.c.

References compress_head_t.

5.241.3.8 compress_t* compress_bzip (stringer_t * *input*)

Compress data using the bzip engine.

Parameters:

input a managed string containing the data to be compressed.

Returns:

NULL on failure, or a pointer to the head of the compressed data on success.

Definition at line 100 of file bzip.c.

References BZ2_bzBuffToBuffCompress_d, compress_alloc(), compress_body_data(), COMPRESS_ENGINE_BZIP, compress_free(), compress_head_t, decompress_bzip(), hash_adler32(), log_info, st_data_get(), st_free(), and st_length_get().

Referenced by engine_compress(), and tank_store().

5.241.3.9 void compress_cleanup (compress_t * *buffer*)

A checked cleanup function which can be used free a compressed object.

Parameters:

buffer a pointer to the head of the compressed object to be freed.

Returns:

This function returns no value.

Definition at line 161 of file compress.c.

References mm_free().

Referenced by mail_store_message().

5.241.3.10 void compress_free (compress_t * *buffer*)

Free a compressed object.

Parameters:

buffer a pointer to the head of the compressed object to be freed.

Returns:

This function returns no value.

Definition at line 150 of file compress.c.

References mm_free().

Referenced by compress_bzip(), compress_lzo(), compress_zlib(), and tank_store().

5.241.3.11 compress_t* compress_import (stringer_t * *s*)

Parse and validate a managed string with compressed data and return a compressed header.

Parameters:

s the managed string containing the compressed data.

Returns:

NULL on general failure or if the Adler 32 hash doesn't match, or a pointer to the data's compressed header on success.

Definition at line 100 of file compress.c.

References compress_body_data(), compress_head_t, hash_adler32(), log_pedantic, st_data_get(), st_empty, and st_length_get().

Referenced by mail_load_message().

5.241.3.12 compress_t* compress_lzo (stringer_t * *input*)

Compress data using the lzo engine.

Parameters:

input a managed string containing the data to be compressed.

Returns:

NULL on failure, or a pointer to the head of the compressed data on success.

Definition at line 132 of file lzo.c.

References compress_alloc(), compress_body_data(), COMPRESS_ENGINE_LZO, compress_free(), compress_head_t, decompress_lzo(), hash_adler32(), log_info, lzo1x_1_compress_d, mm_alloc(), mm_free(), st_data_get(), st_free(), and st_length_get().

Referenced by engine_compress(), mail_store_message(), and tank_store().

5.241.3.13 uint64_t compress_orig_hash (compress_t * *buffer*)

Get the hash value of a compressed buffer's original data.

Parameters:

buffer a pointer to the head of the compressed data.

Returns:

the hash value of the original (uncompressed) data.

Definition at line 34 of file compress.c.

References compress_head_t.

5.241.3.14 uint64_t compress_orig_length (compress_t * *buffer*)

Get the original length of a compressed buffer's data.

Parameters:

buffer a pointer to the head of the compressed data.

Returns:

the original length of the (uncompressed) data.

Definition at line 46 of file compress.c.

References compress_head_t.

5.241.3.15 `uint64_t compress_total_length (compress_t * buffer)`

Return the total length of a compressed header and body.

Returns:

the total length, in bytes, of the compressed header and body.

Definition at line 24 of file `compress.c`.

References `compress_body_length()`, and `compress_head_t`.

Referenced by `mail_store_message()`, and `tank_store()`.

5.241.3.16 `compress_t* compress_zlib (stringer_t * input)`

Compress a block of data using the zlib engine.

Parameters:

input a managed string containing the data to be compressed.

Returns:

NULL on failure, or a pointer to the compressed data header on success..

Definition at line 83 of file `zlib.c`.

References `compress2_d`, `compress_alloc()`, `compress_body_data()`, `COMPRESS_ENGINE_ZLIB`, `compress_free()`, `compress_head_t`, `compressBound_d`, `decompress_zlib()`, `hash_adler32()`, `log_info`, `st_data_get()`, `st_free()`, and `st_length_get()`.

Referenced by `engine_compress()`, and `tank_store()`.

5.241.3.17 `stringer_t* decompress_block_lzo (stringer_t * block)`

Decompress a single block of data using the lzo engine.

Parameters:

block a managed string containing the block of compressed data.

Returns:

NULL on failure, or a managed string containing the uncompressed data on success.

Definition at line 46 of file `lzo.c`.

References `log_info`, `lzo1x_decompress_safe_d`, `st_alloc()`, `st_data_get()`, `st_free()`, `st_length_get()`, and `st_length_set()`.

5.241.3.18 `stringer_t* decompress_bzip (compress_t * compressed)`

Decompress data using the bzip engine.

Parameters:

compressed a pointer to the head of the compressed data.

Returns:

NULL on failure, or a managed string containing the uncompressed data on success.

Definition at line 61 of file bzip.c.

References BZ2_bzBuffToBuffDecompress_d, compress_body_data(), COMPRESS_ENGINE_BZIP, compress_head_t, hash_adler32(), log_info, st_alloc(), st_data_get(), st_free(), and st_length_set().

Referenced by compress_bzip(), engine_decompress(), and tank_load().

5.241.3.19 stringer_t* decompress_lzo (compress_t * *compressed*)

Decompress data using the lzo engine.

Parameters:

compressed a pointer to the head of the compressed data.

Returns:

NULL on failure, or a managed string containing the uncompressed data on success.

Definition at line 86 of file lzo.c.

References compress_body_data(), COMPRESS_ENGINE_LZO, compress_head_t, hash_adler32(), log_info, lzo1x_decompress_safe_d, st_alloc(), st_data_get(), st_free(), and st_length_set().

Referenced by compress_lzo(), engine_decompress(), mail_load_message(), and tank_load().

5.241.3.20 stringer_t* decompress_zlib (compress_t * *compressed*)

Decompress a block of data using the zlib engine.

Parameters:

compressed a pointer to the head of the compressed data.

Returns:

NULL on failure (e.g. corruption), or a managed string containing the uncompressed data on success.

Definition at line 44 of file zlib.c.

References compress_body_data(), COMPRESS_ENGINE_ZLIB, compress_head_t, hash_adler32(), log_info, st_alloc(), st_data_get(), st_free(), st_length_set(), and uncompress_d.

Referenced by compress_zlib(), engine_decompress(), and tank_load().

5.241.3.21 compress_t* engine_compress (uint8_t *engine*, stringer_t * *s*)

[engine.c](#)

Definition at line 10 of file engine.c.

References compress_bzip(), COMPRESS_ENGINE_BZIP, COMPRESS_ENGINE_LZO, COMPRESS_ENGINE_ZLIB, compress_lzo(), compress_zlib(), and log_pedantic.

5.241.3.22 stringer_t* engine_decompress (compress_t * *buffer*)

Definition at line 35 of file engine.c.

References COMPRESS_ENGINE_BZIP, COMPRESS_ENGINE_LZO, COMPRESS_ENGINE_ZLIB, compress_head_t, decompress_bzip(), decompress_lzo(), decompress_zlib(), and log_pedantic.

5.241.3.23 `bool_t lib_load_bzip (void)`

[bzip.c bzip.c](#)

Returns:

true on success or false on failure.

Definition at line 24 of file `bzip.c`.

References `BZ2_bzlibVersion_d`, `bzip_version`, `lib_symbols()`, `log_pedantic`, `M_BIND`, `ns_length_int()`, and `symbol_t`.

Referenced by `lib_load()`.

5.241.3.24 `bool_t lib_load_lzo (void)`

[lzo.c lzo.c](#)

Returns:

true on success or false on failure.

Definition at line 15 of file `lzo.c`.

References `__lzo_init_v2_d`, `lib_symbols()`, `log_critical`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.241.3.25 `bool_t lib_load_zlib (void)`

[zlib.c zlib.c](#)

Returns:

true on success or false on failure.

Definition at line 26 of file `zlib.c`.

References `lib_symbols()`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.241.3.26 `const char* lib_version_bzip (void)`

Return the version string of the bzip library.

Returns:

a pointer to a character string containing the bzip library version information.

Definition at line 16 of file `bzip.c`.

References `bzip_version`.

Referenced by `lib_load()`.

5.241.3.27 `const char* lib_version_lzo (void)`

Return the version string of the lzo library.

Returns:

a pointer to a character string containing the lzo library version information.

Definition at line 37 of file lzo.c.

References lzo_version_string_d.

Referenced by lib_load().

5.241.3.28 const char* lib_version_zlib (void)

Return the version string of zlib.

Returns:

a pointer to a character string containing the zlib version information.

Definition at line 18 of file zlib.c.

References zlibVersion_d.

Referenced by lib_load().

5.241.4 Variable Documentation**5.241.4.1 enum { ... } COMPRESS_ENGINE****5.241.4.2 compress_head_t**

Definition at line 31 of file compress.h.

Referenced by compress_alloc(), compress_body_data(), compress_body_hash(), compress_body_length(), compress_body_offset(), compress_bzip(), compress_import(), compress_lzo(), compress_orig_hash(), compress_orig_length(), compress_total_length(), compress_zlib(), decompress_bzip(), decompress_lzo(), decompress_zlib(), and engine_decompress().

5.242 src/providers/compress/engine.c File Reference

```
#include "magma.h"
```

Functions

- [compress_t * engine_compress](#) (uint8_t engine, [stringer_t *s](#))
[engine.c](#)
- [stringer_t * engine_decompress](#) ([compress_t *buffer](#))

5.242.1 Function Documentation

5.242.1.1 [compress_t * engine_compress](#) (uint8_t *engine*, [stringer_t * s](#))

[engine.c](#)

Definition at line 10 of file engine.c.

References [compress_bzip\(\)](#), [COMPRESS_ENGINE_BZIP](#), [COMPRESS_ENGINE_LZO](#), [COMPRESS_ENGINE_ZLIB](#), [compress_lzo\(\)](#), [compress_zlib\(\)](#), and [log_pedantic](#).

5.242.1.2 [stringer_t * engine_decompress](#) ([compress_t * buffer](#))

Definition at line 35 of file engine.c.

References [COMPRESS_ENGINE_BZIP](#), [COMPRESS_ENGINE_LZO](#), [COMPRESS_ENGINE_ZLIB](#), [compress_head_t](#), [decompress_bzip\(\)](#), [decompress_lzo\(\)](#), [decompress_zlib\(\)](#), and [log_pedantic](#).

5.243 src/providers/compress/lzo.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t lib_load_lzo` (void)
Initialize the lzo library and bind dynamically to the exported functions that are required.
- `const char * lib_version_lzo` (void)
Return the version string of the lzo library.
- `stringer_t * decompress_block_lzo` (stringer_t *block)
Decompress a single block of data using the lzo engine.
- `stringer_t * decompress_lzo` (compress_t *compressed)
Decompress data using the lzo engine.
- `compress_t * compress_lzo` (stringer_t *input)
Compress data using the lzo engine.

5.243.1 Function Documentation

5.243.1.1 `compress_t* compress_lzo` (stringer_t * *input*)

Compress data using the lzo engine.

Parameters:

input a managed string containing the data to be compressed.

Returns:

NULL on failure, or a pointer to the head of the compressed data on success.

Definition at line 132 of file lzo.c.

References `compress_alloc()`, `compress_body_data()`, `COMPRESS_ENGINE_LZO`, `compress_free()`, `compress_head_t`, `decompress_lzo()`, `hash_adler32()`, `log_info`, `lzo1x_l_compress_d`, `mm_alloc()`, `mm_free()`, `st_data_get()`, `st_free()`, and `st_length_get()`.

Referenced by `engine_compress()`, `mail_store_message()`, and `tank_store()`.

5.243.1.2 `stringer_t* decompress_block_lzo` (stringer_t * *block*)

Decompress a single block of data using the lzo engine.

Parameters:

block a managed string containing the block of compressed data.

Returns:

NULL on failure, or a managed string containing the uncompressed data on success.

Definition at line 46 of file lzo.c.

References `log_info`, `lzo1x_decompress_safe_d`, `st_alloc()`, `st_data_get()`, `st_free()`, `st_length_get()`, and `st_length_set()`.

5.243.1.3 `stringer_t* decompress_lzo (compress_t * compressed)`

Decompress data using the lzo engine.

Parameters:

compressed a pointer to the head of the compressed data.

Returns:

NULL on failure, or a managed string containing the uncompressed data on success.

Definition at line 86 of file lzo.c.

References `compress_body_data()`, `COMPRESS_ENGINE_LZO`, `compress_head_t`, `hash_adler32()`, `log_info`, `lzo1x_decompress_safe_d`, `st_alloc()`, `st_data_get()`, `st_free()`, and `st_length_set()`.

Referenced by `compress_lzo()`, `engine_decompress()`, `mail_load_message()`, and `tank_load()`.

5.243.1.4 `bool_t lib_load_lzo (void)`

Initialize the lzo library and bind dynamically to the exported functions that are required. [lzo.c](#)

Returns:

true on success or false on failure.

Definition at line 15 of file lzo.c.

References `__lzo_init_v2_d`, `lib_symbols()`, `log_critical`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.243.1.5 `const char* lib_version_lzo (void)`

Return the version string of the lzo library.

Returns:

a pointer to a character string containing the lzo library version information.

Definition at line 37 of file lzo.c.

References `lzo_version_string_d`.

Referenced by `lib_load()`.

5.244 src/providers/compress/zlib.c File Reference

```
#include "magma.h"
```

Functions

- `const char * lib_version_zlib` (void)
Return the version string of zlib.
- `bool_t lib_load_zlib` (void)
Initialize the zlib library and bind dynamically to the exported functions that are required.
- `stringer_t * decompress_zlib` (`compress_t *compressed`)
Decompress a block of data using the zlib engine.
- `compress_t * compress_zlib` (`stringer_t *input`)
Compress a block of data using the zlib engine.

Variables

- `int(* deflateEnd_d)(z_stream strm) = NULL`
- `int(* deflate_d)(z_stream strm, int flush) = NULL`
- `int(* deflateInit2__d)(z_stream strm, int level, int method, int windowBits, int memLevel, int strategy, const char *version, int stream_size) = NULL`

5.244.1 Function Documentation

5.244.1.1 `compress_t* compress_zlib` (`stringer_t * input`)

Compress a block of data using the zlib engine.

Parameters:

input a managed string containing the data to be compressed.

Returns:

NULL on failure, or a pointer to the compressed data header on success..

Definition at line 83 of file `zlib.c`.

References `compress2_d`, `compress_alloc()`, `compress_body_data()`, `COMPRESS_ENGINE_ZLIB`, `compress_free()`, `compress_head_t`, `compressBound_d`, `decompress_zlib()`, `hash_adler32()`, `log_info`, `st_data_get()`, `st_free()`, and `st_length_get()`.

Referenced by `engine_compress()`, and `tank_store()`.

5.244.1.2 `stringer_t* decompress_zlib` (`compress_t * compressed`)

Decompress a block of data using the zlib engine.

Parameters:

compressed a pointer to the head of the compressed data.

Returns:

NULL on failure (e.g. corruption), or a managed string containing the uncompressed data on success.

Definition at line 44 of file zlib.c.

References `compress_body_data()`, `COMPRESS_ENGINE_ZLIB`, `compress_head_t`, `hash_adler32()`, `log_info`, `st_alloc()`, `st_data_get()`, `st_free()`, `st_length_set()`, and `uncompress_d`.

Referenced by `compress_zlib()`, `engine_decompress()`, and `tank_load()`.

5.244.1.3 bool_t lib_load_zlib (void)

Initialize the zlib library and bind dynamically to the exported functions that are required. [zlib.c](#)

Returns:

true on success or false on failure.

Definition at line 26 of file zlib.c.

References `lib_symbols()`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.244.1.4 const char* lib_version_zlib (void)

Return the version string of zlib.

Returns:

a pointer to a character string containing the zlib version information.

Definition at line 18 of file zlib.c.

References `zlibVersion_d`.

Referenced by `lib_load()`.

5.244.2 Variable Documentation

5.244.2.1 `int(* deflate_d)(z_stream strm, int flush) = NULL`

5.244.2.2 `int(* deflateEnd_d)(z_stream strm) = NULL`

5.244.2.3 `int(* deflateInit2__d)(z_stream strm, int level, int method, int windowBits, int memLevel, int strategy, const char *version, int stream_size) = NULL`

5.245 src/providers/consumers/consumers.h File Reference

Functions

- struct `__attribute__((packed))`
- `int_t cache_add (stringer_t *key, stringer_t *object, time_t expiration)`
cache.c
- `int_t cache_append (stringer_t *key, stringer_t *object, time_t expiration)`
Append data to the value of a cached key on a memcached server.
- `uint64_t cache_decrement (stringer_t *key, uint64_t offset, uint64_t initial, time_t expiration)`
Decrement the value stored with a key inside memcached, using the specified offset.
- `int_t cache_delete (stringer_t *key)`
Delete a key from memcached immediately.
- void `cache_flush (void)`
- `stringer_t * cache_get (stringer_t *key)`
Retrieve data from memcached by key.
- `uint64_t cache_get_u64 (stringer_t *key)`
Retrieve a 64 bit value from memcached by key.
- `uint64_t cache_increment (stringer_t *key, uint64_t offset, uint64_t initial, time_t expiration)`
Increment the value stored with a key by memcached by a specified offset.
- `int_t cache_set (stringer_t *key, stringer_t *object, time_t expiration)`
Set a value in memcached by key.
- `int_t cache_set_u64 (stringer_t *key, uint64_t value, time_t expiration)`
Set a 64 bit value in memcached by key.
- `int_t cache_silent_add (stringer_t *key, stringer_t *object, time_t expiration)`
Add a new key/value pair to the memcached server, but suppress any error messages.
- `bool_t cache_start (void)`
- void `cache_stop (void)`
Destroy the memcached client pool and all active connections.
- `bool_t lib_load_cache (void)`
Initialize the memcached library and bind dynamically to the exported functions that are required.
- const char * `lib_version_cache (void)`
Return the version string of the memcached library.
- `bool_t serialize_sz (stringer_t **data, size_t number)`
Serialization.
- `bool_t serialize_ssz (stringer_t **data, ssize_t number)`
Serialize an ssize_t into a managed string.
- `bool_t serialize_int16 (stringer_t **data, int16_t number)`

Serialize a signed 16-bit integer into a managed string.

- `bool_t serialize_int32 (stringer_t **data, int32_t number)`
Serialize a signed 32-bit integer into a managed string.
- `bool_t serialize_int64 (stringer_t **data, int64_t number)`
Serialize a signed 64-bit integer into a managed string.
- `bool_t serialize_st (stringer_t **data, stringer_t *string)`
Serialize the contents of a managed string into another managed string.
- `bool_t serialize_uint64 (stringer_t **data, int64_t number)`
Serialize an unsigned 64-bit integer into a managed string.
- `bool_t serialize_uint16 (stringer_t **data, uint16_t number)`
Serialize an unsigned 16-bit integer into a managed string.
- `bool_t serialize_uint32 (stringer_t **data, uint32_t number)`
Serialize an unsigned 32-bit integer into a managed string.
- `bool_t serialize_ns (stringer_t **data, char *string, size_t length)`
Serialize a block of memory into a managed string.
- `int_t deserialize_count_digits (chr_t *data, size_t remaining)`
Count the number of bytes before a semicolon or non-numeric character is found.
- `bool_t deserialize_sz (serialization_t *serial, size_t *number)`
Deserialize an size_t from a serialized object, and update its position.
- `bool_t deserialize_ssz (serialization_t *serial, ssize_t *number)`
Deserialize an ssize_t from a serialized object, and update its position.
- `bool_t deserialize_int32 (serialization_t *serial, int_t *number)`
Deserialize a signed 32-bit integer from a serialized object, and update its position.
- `bool_t deserialize_int64 (serialization_t *serial, long int *number)`
- `bool_t deserialize_st (serialization_t *serial, stringer_t **string)`
Deserialize a managed string from a serialized object, and update its position.
- `bool_t deserialize_int16 (serialization_t *serial, short int *number)`
- `bool_t deserialize_uint32 (serialization_t *serial, uint32_t *number)`
Deserialize an unsigned 32-bit integer from a serialized object, and update its position.
- `bool_t deserialize_uint64 (serialization_t *serial, uint64_t *number)`
Deserialize an unsigned 64-bit integer from a serialized object, and update its position.
- `bool_t deserialize_ns (serialization_t *serial, chr_t **string, size_t *length)`
Deserialize a null-terminated string from a serialized object, and update its position.
- `bool_t deserialize_uint16 (serialization_t *serial, short unsigned int *number)`

Variables

- `serialization_t`

5.245.1 Function Documentation

5.245.1.1 struct __attribute__((packed)) [read]

Definition at line 12 of file consumers.h.

References data.

5.245.1.2 int_t cache_add (stringer_t * key, stringer_t * object, time_t expiration)

cache.c cache.c

Parameters:

key a managed string with the name of the key to be added to the cache.

object a managed string containing the initial value of the new key.

expiration an expiration time, in seconds, for the newly cached data.

Definition at line 278 of file cache.c.

References cache_pool, log_info, memcached_add_d, memcached_strerror_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

5.245.1.3 int_t cache_append (stringer_t * key, stringer_t * object, time_t expiration)

Append data to the value of a cached key on a memcached server.

Parameters:

key a managed string with the name of the target memcached key.

object a managed string containing the value of the data being appended to the original key.

expiration an expiration time, in seconds, for the modified cached data.

Definition at line 323 of file cache.c.

References cache_pool, log_info, memcached_add_d, memcached_append_d, memcached_strerror_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by stamp_counter_increment().

5.245.1.4 uint64_t cache_decrement (stringer_t * key, uint64_t offset, uint64_t initial, time_t expiration)

Decrement the value stored with a key inside memcached, using the specified offset.

See also:

memcached_decrement_with_initial()

Note:

If the expiration valuse is MEMCACHED_EXPIRATION_NOT_ADD the key won't be added, but the initial value will be used as the return value.

Parameters:

key a managed string containing the name of the memcached key to be adjusted.

offset the offset by which the specified key's value should be decremented.

initial the initial value to seed the key with it does not already exist.

expiration the time, in seconds, when the cached key will expire.

Returns:

0 on failure, or the newly decremented value associated with the key, or the initial value, on success.

Definition at line 415 of file cache.c.

References `cache_pool`, `log_pedantic`, `memcached_decrement_with_initial_d`, `memcached_strerror_d`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

Referenced by `imap_login()`, `pop_pass()`, `smtp_auth_login()`, and `smtp_auth_plain()`.

5.245.1.5 `int_t cache_delete (stringer_t * key)`

Delete a key from memcached immediately.

Note:

`memcached_delete()`

Parameters:

key the name of the key to be deleted from the memcached server.

Returns:

0 on failure, or `MEMCACHED_SUCCESS` on success.

Definition at line 353 of file cache.c.

References `cache_pool`, `log_info`, `memcached_delete_d`, `memcached_strerror_d`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `st_char_get()`, `st_empty`, `st_length_get()`, and `st_length_int()`.

Referenced by `lock_release()`.

5.245.1.6 `void cache_flush (void)`

Definition at line 124 of file cache.c.

References `cache_pool`, `log_info`, `memcached_flush_d`, `memcached_strerror_d`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, and `pool_release()`.

5.245.1.7 `stringer_t* cache_get (stringer_t * key)`

Retrieve data from memcached by key.

Parameters:

key a managed string containing a key to be passed to memcached.

Returns:

NULL on failure, or a pointer to a managed string containing the cached data on success.

Definition at line 145 of file cache.c.

References `cache_pool`, `data`, `length`, `log_info`, `memcached_get_d`, `memcached_strerror_d`, `mm_free()`, `PL_RESERVED`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `st_char_get()`, `st_empty`, `st_import()`, `st_length_get()`, and `st_length_int()`.

Referenced by `register_session_get()`, `smtp_check_greylist()`, `smtp_fetch_autoreply()`, `stamp_counter_check()`, and `teacher_data_get()`.

5.245.1.8 uint64_t cache_get_u64 (stringer_t * key)

Retrieve a 64 bit value from memcached by key.

Parameters:

key a managed string containing a key to be passed to memcached.

Returns:

NULL on failure, or the 64 bit value cached data on success.

Definition at line 185 of file cache.c.

References cache_pool, data, length, log_info, memcached_get_d, memcached_strerror_d, mm_free(), PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by contact_abuse_checks(), register_abuse_checks(), and smtp_reply().

5.245.1.9 uint64_t cache_increment (stringer_t * key, uint64_t offset, uint64_t initial, time_t expiration)

Increment the value stored with a key by memcached by a specified offset.

See also:

memcached_increment_with_initial()

Note:

If the expiration value is MEMCACHED_EXPIRATION_NOT_ADD the key won't be added, but the initial value will be used as the return value.

Parameters:

key a managed string containing the name of the memcached key to be adjusted.

offset the offset by which the specified key's value should be incremented.

initial the initial value to seed the key if it does not already exist.

expiration the time, in seconds, when the cached key will expire.

Returns:

0 on failure, or the newly incremented value associated with the key, or the initial value, on success.

Definition at line 381 of file cache.c.

References cache_pool, log_pedantic, memcached_increment_with_initial_d, memcached_strerror_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by api_endpoint_auth(), contact_abuse_increment_history(), imap_login(), pop_pass(), portal_endpoint_auth(), register_abuse_increment_history(), serial_get(), serial_increment(), smtp_auth_login(), and smtp_auth_plain().

5.245.1.10 int_t cache_set (stringer_t * key, stringer_t * object, time_t expiration)

Set a value in memcached by key.

Parameters:

key a managed string containing a key to be passed to memcached.

object a managed string containing the new data to be associated with the supplied key.

expiration the expiration time of the cached data.

Returns:

0 on failure, or 1 on success.

Definition at line 231 of file cache.c.

References cache_pool, log_info, memcached_set_d, memcached_strerror_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by register_session_cache(), serial_reset(), smtp_check_greylist(), smtp_fetch_autoreply(), and teacher_data_save().

5.245.1.11 int_t cache_set_u64 (stringer_t * key, uint64_t value, time_t expiration)

Set a 64 bit value in memcached by key.

Parameters:

key a managed string containing a key to be passed to memcached.

value the new value to be associated with the supplied key.

expiration the expiration time of the cached data.

Returns:

NULL on failure, or the retrieved unsigned 64 bit cached data on success.

Definition at line 255 of file cache.c.

References cache_pool, log_info, memcached_set_d, memcached_strerror_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, st_length_get(), and st_length_int().

Referenced by smtp_reply().

5.245.1.12 int_t cache_silent_add (stringer_t * key, stringer_t * object, time_t expiration)

Add a new key/value pair to the memcached server, but suppress any error messages.

Parameters:

key a managed string with the name of the key to be added to the cache.

object a managed string containing the initial value of the new key.

expiration an expiration time, in seconds, for the newly cached data.

Definition at line 301 of file cache.c.

References cache_pool, memcached_add_d, PL_RESERVED, pool_get_obj(), pool_pull(), pool_release(), st_char_get(), st_empty, and st_length_get().

Referenced by lock_get().

5.245.1.13 bool_t cache_start (void)

Definition at line 45 of file cache.c.

References magma_t::cache, cache_pool, magma_t::iface, log_critical, magma, MAGMA_CACHE_INSTANCES, memcached_behavior_set_d, memcached_create_d, memcached_free_d, memcached_server_add_with_weight_d, memcached_strerror_d, pool_alloc(), and pool_set_obj().

Referenced by process_start().

5.245.1.14 void cache_stop (void)

Destroy the memcached client pool and all active connections.

Returns:

This function returns no value.

Definition at line 104 of file cache.c.

References magma_t::cache, cache_pool, magma_t::iface, magma, memcached_free_d, pool_free(), and pool_get_obj().

Referenced by process_stop().

5.245.1.15 int_t deserialize_count_digits (chr_t * *data*, size_t *remaining*) [inline]

Count the number of bytes before a semicolon or non-numeric character is found.

Parameters:

data a pointer to a null-terminated string to be scanned.

remaining the maximum number of characters to be scanned.

Returns:

-1 if no match is found, or the length, in bytes, of the data preceding the matched character.

Definition at line 16 of file deserialization.c.

Referenced by deserialize_int16(), deserialize_int32(), deserialize_int64(), deserialize_uint16(), deserialize_uint32(), and deserialize_uint64().

5.245.1.16 bool_t deserialize_int16 (serialization_t * *serial*, short int * *number*)

5.245.1.17 bool_t deserialize_int32 (serialization_t * *serial*, int_t * *number*)

Deserialize a signed 32-bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to a signed 32-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 138 of file deserialization.c.

References deserialize_count_digits(), int32_conv_bl(), length, log_pedantic, st_char_get(), and st_length_get().

Referenced by teacher_data_get().

5.245.1.18 bool_t deserialize_int64 (serialization_t * *serial*, long int * *number*)

Referenced by deserialize_ssz().

5.245.1.19 `bool_t deserialize_ns(serialization_t * serial, chr_t ** string, size_t * length)`

Deserialize a null-terminated string from a serialized object, and update its position.

Note:

A null-terminated string is stored as a 64 bit length field, followed by the string contents.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

string a pointer to the address of a null-terminated string that will receive the de-serialized data at the serialized object's current position.

length a pointer to a size_t that will receive the length of the de-serialized null-terminated string on success.

Returns:

true on success or false on failure..

Definition at line 352 of file deserialization.c.

References bytes, deserialize_sz(), log_pedantic, mm_copy(), ns_alloc(), st_char_get(), and st_length_get().

5.245.1.20 `bool_t deserialize_ssz(serialization_t * serial, ssize_t * number)`

Deserialize an ssize_t from a serialized object, and update its position.

See also:

[deserialize_int64](#)

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an ssize_t variable which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 48 of file deserialization.c.

References deserialize_int64().

5.245.1.21 `bool_t deserialize_st(serialization_t * serial, stringer_t ** string)`

Deserialize a managed string from a serialized object, and update its position.

Note:

A managed string is stored as a 64 bit length field, followed by opaque data.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to the address of a managed string, will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure..

Definition at line 299 of file deserialization.c.

References `deserialize_sz()`, `length`, `log_pedantic`, `mm_copy()`, `st_alloc()`, `st_char_get()`, `st_length_get()`, and `st_length_set()`.

Referenced by `register_session_get()`, and `teacher_data_get()`.

5.245.1.22 `bool_t deserialize_sz(serialization_t * serial, size_t * number)`

Deserialize an `size_t` from a serialized object, and update its position.

See also:

[deserialize_uint64](#)

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an `size_t` variable which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 37 of file deserialization.c.

References `deserialize_uint64()`.

Referenced by `deserialize_ns()`, and `deserialize_st()`.

5.245.1.23 `bool_t deserialize_uint16(serialization_t * serial, short unsigned int * number)`

Referenced by `register_session_get()`.

5.245.1.24 `bool_t deserialize_uint32(serialization_t * serial, uint32_t * number)`

Deserialize an unsigned 32-bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an unsigned 32-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 98 of file deserialization.c.

References `deserialize_count_digits()`, `length`, `log_pedantic`, `st_char_get()`, `st_length_get()`, and `uint32_conv_bl()`.

5.245.1.25 `bool_t deserialize_uint64(serialization_t * serial, uint64_t * number)`

Deserialize an unsigned 64-bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an unsigned 64-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 58 of file deserialization.c.

References `deserialize_count_digits()`, `length`, `log_pedantic`, `st_char_get()`, `st_length_get()`, and `uint64_conv_bl()`.

Referenced by `deserialize_sz()`, `register_session_get()`, and `teacher_data_get()`.

5.245.1.26 bool_t lib_load_cache (void)

Initialize the memcached library and bind dynamically to the exported functions that are required.

Returns:

true on success or false on failure.

Definition at line 16 of file cache.c.

References `lib_symbols()`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.245.1.27 const char* lib_version_cache (void)

Return the version string of the memcached library.

Returns:

a pointer to a character string containing the memcached library version information.

Definition at line 37 of file cache.c.

References `memcached_lib_version_d`.

Referenced by `lib_load()`.

5.245.1.28 bool_t serialize_int16 (stringer_t ** data, int16_t number)

Serialize a signed 16-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the signed 16-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 39 of file serialization.c.

References `int16_digits()`, `length`, `st_alloc()`, `st_avail_get()`, `st_char_get()`, `st_length_get()`, `st_length_set()`, and `st_realloc()`.

5.245.1.29 bool_t serialize_int32 (stringer_t ** *data*, int32_t *number*)

Serialize a signed 32-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the signed 32-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 111 of file serialization.c.

References int32_digits(), length, st_alloc(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), and st_realloc().

Referenced by teacher_data_save().

5.245.1.30 bool_t serialize_int64 (stringer_t ** *data*, int64_t *number*)

Serialize a signed 64-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the signed 64-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 182 of file serialization.c.

References int64_digits(), length, st_alloc(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), and st_realloc().

Referenced by serialize_ssz().

5.245.1.31 bool_t serialize_ns (stringer_t ** *data*, char * *string*, size_t *length*)

Serialize a block of memory into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

string a pointer to the start of the memory block to be serialized.

length the length, in bytes, of the memory block to be serialized.

Returns:

1 if the block of memory was serialized and stored successfully, or -1 on failure.

Definition at line 302 of file serialization.c.

References mm_copy(), serialize_sz(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), and st_realloc().

5.245.1.32 bool_t serialize_ssz (stringer_t ** data, ssize_t number)

Serialize an ssize_t into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the ssize_t to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 28 of file serialization.c.

References serialize_int64().

5.245.1.33 bool_t serialize_st (stringer_t ** data, stringer_t * string)

Serialize the contents of a managed string into another managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

string a pointer to a managed string containing the data to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 254 of file serialization.c.

References length, mm_copy(), serialize_sz(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), and st_realloc().

Referenced by register_session_cache(), and teacher_data_save().

5.245.1.34 bool_t serialize_sz (stringer_t ** data, size_t number)

Serialization. Serialization.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the size_t to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 17 of file serialization.c.

References serialize_uint64().

Referenced by serialize_ns(), and serialize_st().

5.245.1.35 bool_t serialize_uint16 (stringer_t ** data, uint16_t number)

Serialize an unsigned 16-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the unsigned 16-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 75 of file serialization.c.

References length, st_alloc(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), st_realloc(), and uint16_digits().

Referenced by register_session_cache().

5.245.1.36 bool_t serialize_uint32 (stringer_t ** data, uint32_t number)

Serialize an unsigned 32-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the unsigned 32-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 147 of file serialization.c.

References length, st_alloc(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), st_realloc(), and uint32_digits().

5.245.1.37 bool_t serialize_uint64 (stringer_t ** data, int64_t number)

Serialize an unsigned 64-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the unsigned 64-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 218 of file serialization.c.

References length, st_alloc(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), st_realloc(), and uint64_digits().

Referenced by register_session_cache(), serialize_sz(), and teacher_data_save().

5.245.2 Variable Documentation**5.245.2.1 serialization_t**

Definition at line 15 of file consumers.h.

Referenced by register_session_get(), and teacher_data_get().

5.246 src/providers/consumers/deserialization.c File Reference

```
#include "magma.h"
```

Functions

- `int_t deserialize_count_digits (chr_t *data, size_t remaining)`
Count the number of bytes before a semicolon or non-numeric character is found.
- `bool_t deserialize_sz (serialization_t *serial, size_t *number)`
Deserialize an size_t from a serialized object, and update its position.
- `bool_t deserialize_ssz (serialization_t *serial, ssize_t *number)`
Deserialize an ssize_t from a serialized object, and update its position.
- `bool_t deserialize_uint64 (serialization_t *serial, uint64_t *number)`
Deserialize an unsigned 64-bit integer from a serialized object, and update its position.
- `bool_t deserialize_uint32 (serialization_t *serial, uint32_t *number)`
Deserialize an unsigned 32-bit integer from a serialized object, and update its position.
- `bool_t deserialize_int32 (serialization_t *serial, int_t *number)`
Deserialize a signed 32-bit integer from a serialized object, and update its position.
- `bool_t deserialize_int64 (serialization_t *serial, long *number)`
Deserialize a signed 64-bit integer from a serialized object, and update its position.
- `bool_t deserialize_int16 (serialization_t *serial, short *number)`
Deserialize a signed 16 bit integer from a serialized object, and update its position.
- `bool_t deserialize_uint16 (serialization_t *serial, unsigned short *number)`
Deserialize an unsigned 16 bit integer from a serialized object, and update its position.
- `bool_t deserialize_st (serialization_t *serial, stringer_t **string)`
Deserialize a managed string from a serialized object, and update its position.
- `bool_t deserialize_ns (serialization_t *serial, chr_t **string, size_t *length)`
Deserialize a null-terminated string from a serialized object, and update its position.

5.246.1 Function Documentation

5.246.1.1 int_t deserialize_count_digits (chr_t *data, size_t remaining) [inline]

Count the number of bytes before a semicolon or non-numeric character is found.

Parameters:

- data* a pointer to a null-terminated string to be scanned.
- remaining* the maximum number of characters to be scanned.

Returns:

- 1 if no match is found, or the length, in bytes, of the data preceding the matched character.

Definition at line 16 of file deserialization.c.

Referenced by `deserialize_int16()`, `deserialize_int32()`, `deserialize_int64()`, `deserialize_uint16()`, `deserialize_uint32()`, and `deserialize_uint64()`.

5.246.1.2 `bool_t deserialize_int16 (serialization_t * serial, short * number)`

Deserialize a signed 16 bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to a signed 16-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 218 of file deserialization.c.

References `deserialize_count_digits()`, `int16_conv_bl()`, `length`, `log_pedantic`, `st_char_get()`, and `st_length_get()`.

5.246.1.3 `bool_t deserialize_int32 (serialization_t * serial, int_t * number)`

Deserialize a signed 32-bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to a signed 32-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 138 of file deserialization.c.

References `deserialize_count_digits()`, `int32_conv_bl()`, `length`, `log_pedantic`, `st_char_get()`, and `st_length_get()`.

Referenced by `teacher_data_get()`.

5.246.1.4 `bool_t deserialize_int64 (serialization_t * serial, long * number)`

Deserialize a signed 64-bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to a signed 64-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 178 of file deserialization.c.

References `deserialize_count_digits()`, `int64_conv_bl()`, `length`, `log_pedantic`, `st_char_get()`, and `st_length_get()`.

5.246.1.5 bool_t deserialize_ns (serialization_t * serial, chr_t ** string, size_t * length)

Deserialize a null-terminated string from a serialized object, and update its position.

Note:

A null-terminated string is stored as a 64 bit length field, followed by the string contents.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

string a pointer to the address of a null-terminated string that will receive the de-serialized data at the serialized object's current position.

length a pointer to a size_t that will receive the length of the de-serialized null-terminated string on success.

Returns:

true on success or false on failure..

Definition at line 352 of file deserialization.c.

References bytes, deserialize_sz(), log_pedantic, mm_copy(), ns_alloc(), st_char_get(), and st_length_get().

5.246.1.6 bool_t deserialize_ssz (serialization_t * serial, ssize_t * number)

Deserialize an ssize_t from a serialized object, and update its position.

See also:

[deserialize_int64](#)

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an ssize_t variable which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 48 of file deserialization.c.

References deserialize_int64().

5.246.1.7 bool_t deserialize_st (serialization_t * serial, stringer_t ** string)

Deserialize a managed string from a serialized object, and update its position.

Note:

A managed string is stored as a 64 bit length field, followed by opaque data.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to the address of a managed string, will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure..

Definition at line 299 of file deserialization.c.

References `deserialize_sz()`, `length`, `log_pedantic`, `mm_copy()`, `st_alloc()`, `st_char_get()`, `st_length_get()`, and `st_length_set()`.

Referenced by `register_session_get()`, and `teacher_data_get()`.

5.246.1.8 **bool_t deserialize_sz (serialization_t * *serial*, size_t * *number*)**

Deserialize an `size_t` from a serialized object, and update its position.

See also:

[deserialize_uint64](#)

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an `size_t` variable which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 37 of file deserialization.c.

References `deserialize_uint64()`.

Referenced by `deserialize_ns()`, and `deserialize_st()`.

5.246.1.9 **bool_t deserialize_uint16 (serialization_t * *serial*, unsigned short * *number*)**

Deserialize an unsigned 16 bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an unsigned 16-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure..

Definition at line 258 of file deserialization.c.

References `deserialize_count_digits()`, `length`, `log_pedantic`, `st_char_get()`, `st_length_get()`, and `uint16_conv_bl()`.

5.246.1.10 **bool_t deserialize_uint32 (serialization_t * *serial*, uint32_t * *number*)**

Deserialize an unsigned 32-bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an unsigned 32-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 98 of file deserialization.c.

References `deserialize_count_digits()`, `length`, `log_pedantic`, `st_char_get()`, `st_length_get()`, and `uint32_conv_bl()`.

5.246.1.11 bool_t deserialize_uint64 (serialization_t * *serial*, uint64_t * *number*)

Deserialize an unsigned 64-bit integer from a serialized object, and update its position.

Parameters:

serial a pointer to a serialized object from which the data will be extracted.

number a pointer to an unsigned 64-bit integer which will receive the de-serialized data at the serialized object's current position.

Returns:

true on success or false on failure.

Definition at line 58 of file deserialization.c.

References `deserialize_count_digits()`, `length`, `log_pedantic`, `st_char_get()`, `st_length_get()`, and `uint64_conv_bl()`.

Referenced by `deserialize_sz()`, `register_session_get()`, and `teacher_data_get()`.

5.247 src/providers/consumers/serialization.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t serialize_sz](#) ([stringer_t **data](#), [size_t number](#))
Serialize a size_t into a managed string.
- [bool_t serialize_ssz](#) ([stringer_t **data](#), [ssize_t number](#))
Serialize an ssize_t into a managed string.
- [bool_t serialize_int16](#) ([stringer_t **data](#), [int16_t number](#))
Serialize a signed 16-bit integer into a managed string.
- [bool_t serialize_uint16](#) ([stringer_t **data](#), [uint16_t number](#))
Serialize an unsigned 16-bit integer into a managed string.
- [bool_t serialize_int32](#) ([stringer_t **data](#), [int32_t number](#))
Serialize a signed 32-bit integer into a managed string.
- [bool_t serialize_uint32](#) ([stringer_t **data](#), [uint32_t number](#))
Serialize an unsigned 32-bit integer into a managed string.
- [bool_t serialize_int64](#) ([stringer_t **data](#), [int64_t number](#))
Serialize a signed 64-bit integer into a managed string.
- [bool_t serialize_uint64](#) ([stringer_t **data](#), [int64_t number](#))
Serialize an unsigned 64-bit integer into a managed string.
- [bool_t serialize_st](#) ([stringer_t **data](#), [stringer_t *string](#))
Serialize the contents of a managed string into another managed string.
- [bool_t serialize_ns](#) ([stringer_t **data](#), [char *string](#), [size_t length](#))
Serialize a block of memory into a managed string.

5.247.1 Function Documentation

5.247.1.1 bool_t serialize_int16 (stringer_t ** data, int16_t number)

Serialize a signed 16-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.
number the value of the signed 16-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 39 of file serialization.c.

References `int16_digits()`, `length`, `st_alloc()`, `st_avail_get()`, `st_char_get()`, `st_length_get()`, `st_length_set()`, and `st_realloc()`.

5.247.1.2 `bool_t serialize_int32 (stringer_t ** data, int32_t number)`

Serialize a signed 32-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the signed 32-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 111 of file serialization.c.

References `int32_digits()`, `length`, `st_alloc()`, `st_avail_get()`, `st_char_get()`, `st_length_get()`, `st_length_set()`, and `st_realloc()`.

Referenced by `teacher_data_save()`.

5.247.1.3 `bool_t serialize_int64 (stringer_t ** data, int64_t number)`

Serialize a signed 64-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the signed 64-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 182 of file serialization.c.

References `int64_digits()`, `length`, `st_alloc()`, `st_avail_get()`, `st_char_get()`, `st_length_get()`, `st_length_set()`, and `st_realloc()`.

Referenced by `serialize_ssz()`.

5.247.1.4 `bool_t serialize_ns (stringer_t ** data, char * string, size_t length)`

Serialize a block of memory into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.
string a pointer to the start of the memory block to be serialized.
length the length, in bytes, of the memory block to be serialized.

Returns:

1 if the block of memory was serialized and stored successfully, or -1 on failure.

Definition at line 302 of file serialization.c.

References mm_copy(), serialize_sz(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), and st_realloc().

5.247.1.5 bool_t serialize_ssz (stringer_t ** data, ssize_t number)

Serialize an ssize_t into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.
number the value of the ssize_t to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 28 of file serialization.c.

References serialize_int64().

5.247.1.6 bool_t serialize_st (stringer_t ** data, stringer_t * string)

Serialize the contents of a managed string into another managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.
string a pointer to a managed string containing the data to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 254 of file serialization.c.

References length, mm_copy(), serialize_sz(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), and st_realloc().

Referenced by register_session_cache(), and teacher_data_save().

5.247.1.7 bool_t serialize_sz (stringer_t ** data, size_t number)

Serialize a size_t into a managed string. Serialization.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the size_t to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 17 of file serialization.c.

References serialize_uint64().

Referenced by serialize_ns(), and serialize_st().

5.247.1.8 bool_t serialize_uint16 (stringer_t ** data, uint16_t number)

Serialize an unsigned 16-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the unsigned 16-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 75 of file serialization.c.

References length, st_alloc(), st_avail_get(), st_char_get(), st_length_get(), st_length_set(), st_realloc(), and uint16_digits().

Referenced by register_session_cache().

5.247.1.9 bool_t serialize_uint32 (stringer_t ** data, uint32_t number)

Serialize an unsigned 32-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the unsigned 32-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 147 of file serialization.c.

References `length`, `st_alloc()`, `st_avail_get()`, `st_char_get()`, `st_length_get()`, `st_length_set()`, `st_realloc()`, and `uint32_digits()`.

5.247.1.10 `bool_t serialize_uint64 (stringer_t **data, int64_t number)`

Serialize an unsigned 64-bit integer into a managed string.

Note:

This function will reallocate the output managed string if necessary, and append the serialized data, updating the length field to reflect the changes.

Parameters:

data a pointer to the address of a managed string to receive the output, which will be allocated for the caller if NULL is passed.

number the value of the unsigned 64-bit integer to be serialized.

Returns:

true if the specified value was serialized and stored successfully, or false on failure.

Definition at line 218 of file serialization.c.

References `length`, `st_alloc()`, `st_avail_get()`, `st_char_get()`, `st_length_get()`, `st_length_set()`, `st_realloc()`, and `uint64_digits()`.

Referenced by `register_session_cache()`, `serialize_sz()`, and `teacher_data_save()`.

5.248 src/providers/cryptography/ciphers.c File Reference

```
#include "magma.h"
```

Functions

- `cipher_t * cipher_name (stringer_t *name)`
Look up a cipher type by name.
- `cipher_t * cipher_id (int_t id)`
Look up a cipher type by its nid.
- `int_t cipher_numeric_id (cipher_t *c)`
Return the numerical ID (nid) of a specified cipher.
- `int_t cipher_key_length (cipher_t *c)`
Determine the key length of a specified cipher.
- `int_t cipher_vector_length (cipher_t *c)`
Determine the IV length of a specified cipher.
- `int_t cipher_block_length (cipher_t *c)`
Determine the block length of a specified cipher.

5.248.1 Function Documentation

5.248.1.1 `int_t cipher_block_length (cipher_t * c)`

Determine the block length of a specified cipher.

Parameters:

c the input cipher type.

Returns:

-1 on failure, or the cipher's block length in bytes.

Definition at line 94 of file ciphers.c.

References `EVP_CIPHER_block_size_d`, `EVP_CIPHER_nid_d`, `log_pedantic`, and `OBJ_nid2sn_d`.

5.248.1.2 `cipher_t* cipher_id (int_t id)`

Look up a cipher type by its nid. [ciphers.c](#)

Parameters:

id the numerical ID (nid) of the specified cipher.

Returns:

NULL on failure, or the relevant cipher data structure on success.

Definition at line 30 of file ciphers.c.

References `EVP_get_cipherbyname_d`, `log_pedantic`, and `OBJ_nid2sn_d`.

Referenced by `deprecated_scramble_decrypt()`, `deprecated_scramble_encrypt()`, `scramble_decrypt()`, and `scramble_encrypt()`.

5.248.1.3 `int_t cipher_key_length (cipher_t * c)`

Determine the key length of a specified cipher.

Parameters:

c the input cipher type.

Returns:

-1 on failure, or the cipher's key length in bytes.

Definition at line 62 of file ciphers.c.

References `EVP_CIPHER_key_length_d`, `EVP_CIPHER_nid_d`, `log_pedantic`, and `OBJ_nid2sn_d`.

Referenced by `deprecated_symmetric_key()`, and `symmetric_key()`.

5.248.1.4 `cipher_t* cipher_name (stringer_t * name)`

Look up a cipher type by name.

Parameters:

name a descriptive name of the cipher to be looked up.

Returns:

NULL on failure, or the relevant cipher data structure on success.

Definition at line 15 of file ciphers.c.

References `EVP_get_cipherbyname_d`, `log_pedantic`, `st_char_get()`, `st_empty`, and `st_length_int()`.

5.248.1.5 `int_t cipher_numeric_id (cipher_t * c)`

Return the numerical ID (nid) of a specified cipher.

Parameters:

c the input cipher type.

Returns:

NID_undef on failure, or the nid of the specified cipher.

Definition at line 46 of file ciphers.c.

References `EVP_CIPHER_nid_d`, and `log_pedantic`.

Referenced by `deprecated_scramble_encrypt()`, and `scramble_encrypt()`.

5.248.1.6 int_t cipher_vector_length (cipher_t * c)

Determine the IV length of a specified cipher.

Parameters:

c the input cipher type.

Returns:

-1 on failure, or the cipher's IV length in bytes.

Definition at line 78 of file ciphers.c.

References `EVP_CIPHER_iv_length_d`, `EVP_CIPHER_nid_d`, `log_pedantic`, and `OBJ_nid2sn_d`.

Referenced by `deprecated_symmetric_vector()`, and `symmetric_vector()`.

5.249 src/providers/cryptography/cryptex.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t deprecated_cryptex_envelope_length (cryptex_t *cryptex)`
Get the length of a cryptex envelope.
- `uint64_t deprecated_cryptex_hmac_length (cryptex_t *cryptex)`
Get the length of a cryptex HMAC.
- `uint64_t deprecated_cryptex_body_length (cryptex_t *cryptex)`
Get the length of a cryptex object's encrypted data body.
- `uint64_t deprecated_cryptex_original_length (cryptex_t *cryptex)`
Get the original length of a cryptex object's data buffer.
- `uint64_t deprecated_cryptex_total_length (cryptex_t *cryptex)`
Determine the total length of the data underlying a cryptex object.
- `void * deprecated_cryptex_envelope_data (cryptex_t *cryptex)`
Get a pointer to the envelope data of a cryptex object.
- `void * deprecated_cryptex_hmac_data (cryptex_t *cryptex)`
Get a pointer to the HMAC data of a cryptex object.
- `void * deprecated_cryptex_body_data (cryptex_t *cryptex)`
Get a pointer to the encrypted data body of a cryptex object.
- `void * deprecated_cryptex_alloc (uint64_t envelope, uint64_t hmac, uint64_t original, uint64_t body)`
Allocate a new cryptex object.
- `void deprecated_cryptex_free (cryptex_t *cryptex)`
Free a cryptex object.

5.249.1 Function Documentation

5.249.1.1 `void* deprecated_cryptex_alloc (uint64_t envelope, uint64_t hmac, uint64_t original, uint64_t body)`

Allocate a new cryptex object.

Definition at line 108 of file cryptex.c.

References `cryptex_head_t`, `log_pedantic`, and `mm_alloc()`.

Referenced by `deprecated_ecies_encrypt()`.

5.249.1.2 `void* deprecated_cryptex_body_data (cryptex_t * cryptex)`

Get a pointer to the encrypted data body of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's encrypted data body.

Definition at line 97 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.249.1.3 uint64_t deprecated_cryptex_body_length (cryptex_t * cryptex)

Get the length of a cryptex object's encrypted data body. cryptex.c

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex encrypted data body.

Definition at line 39 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.249.1.4 void* deprecated_cryptex_envelope_data (cryptex_t * cryptex)

Get a pointer to the envelope data of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's envelope data.

Definition at line 75 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.249.1.5 uint64_t deprecated_cryptex_envelope_length (cryptex_t * cryptex)

Get the length of a cryptex envelope.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex envelope.

Definition at line 15 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt().

5.249.1.6 void deprecated_cryptex_free (cryptex_t * cryptex)

Free a cryptex object.

Parameters:

cryptex the cryptex object to be freed.

Returns:

This function returns no value.

Definition at line 132 of file cryptex.c.

Referenced by deprecated_ecies_encrypt().

5.249.1.7 void* deprecated_cryptex_hmac_data (cryptex_t * cryptex)

Get a pointer to the HMAC data of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's hmac data.

Definition at line 85 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.249.1.8 uint64_t deprecated_cryptex_hmac_length (cryptex_t * cryptex)

Get the length of a cryptex HMAC.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex HMAC.

Definition at line 27 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.249.1.9 uint64_t deprecated_cryptex_original_length (cryptex_t * cryptex)

Get the original length of a cryptex object's data buffer.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex object's original data.

Definition at line 51 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt().

5.249.1.10 uint64_t deprecated_cryptex_total_length (cryptex_t * cryptex)

Determine the total length of the data underlying a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

the total length of the associated cryptex data in bytes.

Definition at line 63 of file cryptex.c.

References cryptex_head_t.

5.250 src/providers/deprecated/cryptex.c File Reference

```
#include "magma.h"
#include "deprecated.h"
```

Functions

- `uint64_t cryptex_envelope_length (cryptex_t *cryptex)`
Get the length of a cryptex envelope.
- `uint64_t cryptex_hmac_length (cryptex_t *cryptex)`
Get the length of a cryptex HMAC.
- `uint64_t cryptex_body_length (cryptex_t *cryptex)`
Get the length of a cryptex object's encrypted data body.
- `uint64_t cryptex_original_length (cryptex_t *cryptex)`
Get the original length of a cryptex object's data buffer.
- `uint64_t cryptex_total_length (cryptex_t *cryptex)`
Determine the total length of the data underlying a cryptex object.
- `void * cryptex_envelope_data (cryptex_t *cryptex)`
Get a pointer to the envelope data of a cryptex object.
- `void * cryptex_hmac_data (cryptex_t *cryptex)`
Get a pointer to the HMAC data of a cryptex object.
- `void * cryptex_body_data (cryptex_t *cryptex)`
Get a pointer to the encrypted data body of a cryptex object.
- `void * cryptex_alloc (uint64_t envelope, uint64_t hmac, uint64_t original, uint64_t body)`
Allocate a new cryptex object.
- `void cryptex_free (cryptex_t *cryptex)`
Free a cryptex object.

5.250.1 Function Documentation

5.250.1.1 `void* cryptex_alloc (uint64_t envelope, uint64_t hmac, uint64_t original, uint64_t body)`

Allocate a new cryptex object.

Definition at line 109 of file cryptex.c.

References `cryptex_head_t`, `log_pedantic`, and `mm_alloc()`.

Referenced by `ecies_encrypt()`.

5.250.1.2 void* cryptex_body_data (cryptex_t * cryptex)

Get a pointer to the encrypted data body of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's encrypted data body.

Definition at line 98 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.250.1.3 uint64_t cryptex_body_length (cryptex_t * cryptex)

Get the length of a cryptex object's encrypted data body. cryptex.c

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex encrypted data body.

Definition at line 40 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.250.1.4 void* cryptex_envelope_data (cryptex_t * cryptex)

Get a pointer to the envelope data of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's envelope data.

Definition at line 76 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.250.1.5 uint64_t cryptex_envelope_length (cryptex_t * cryptex)

Get the length of a cryptex envelope.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex envelope.

Definition at line 16 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt().

5.250.1.6 void cryptex_free (cryptex_t * cryptex)

Free a cryptex object.

Parameters:

cryptex the cryptex object to be freed.

Returns:

This function returns no value.

Definition at line 133 of file cryptex.c.

Referenced by ecies_encrypt().

5.250.1.7 void* cryptex_hmac_data (cryptex_t * cryptex)

Get a pointer to the HMAC data of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's hmac data.

Definition at line 86 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.250.1.8 uint64_t cryptex_hmac_length (cryptex_t * cryptex)

Get the length of a cryptex HMAC.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex HMAC.

Definition at line 28 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.250.1.9 uint64_t cryptex_original_length (cryptex_t * cryptex)

Get the original length of a cryptex object's data buffer.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex object's original data.

Definition at line 52 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt().

5.250.1.10 uint64_t cryptex_total_length (cryptex_t * cryptex)

Determine the total length of the data underlying a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

the total length of the associated cryptex data in bytes.

Definition at line 64 of file cryptex.c.

References cryptex_head_t.

5.251 src/providers/cryptography/cryptography.h File Reference

Defines

- #define `BN_num_bytes_d(a)` `((BN_num_bits_d(a)+7)/8)`
- #define `OPENSSL_free_d(addr)` `CRYPTO_free_d(addr)`
- #define `MAGMA_CIPHERS_HIGH` "ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305"
- #define `MAGMA_CIPHERS_MEDIUM` "EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA384:EECDH+ECDSA+SHA:!SEED"
- #define `MAGMA_CIPHERS_LOW` "EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA384:EECDH+ECDSA+SHA:!SEED"
- #define `MAGMA_CIPHERS_GENERIC` "HIGH:MEDIUM"
- #define `ECIES_HMAC` `NID_sha512`
- #define `ECIES_CURVE` `NID_sect571k1`
- #define `ECIES_CIPHER` `NID_aes_256_cbc`
- #define `ECIES_ENVELOPE` `NID_sha512`

Typedefs

- typedef void * `digest_t`
- typedef void * `cipher_t`
- typedef char * `cryptex_t`
- typedef `stringer_t` `scramble_t`
- typedef void `TLS`

Enumerations

- enum `ECIES_KEY_TYPE` { `ECIES_PRIVATE_HEX` = 1, `ECIES_PRIVATE_BINARY` = 2, `ECIES_PUBLIC_HEX` = 4, `ECIES_PUBLIC_BINARY` = 8 }

Functions

- struct `__attribute__((packed))`
- `cipher_t * cipher_id(int_t id)`
ciphers.c
- `cipher_t * cipher_name(stringer_t *name)`
Look up a cipher type by name.
- `int_t cipher_numeric_id(cipher_t *c)`
Return the numerical ID (nid) of a specified cipher.
- `int_t cipher_block_length(cipher_t *c)`
Determine the block length of a specified cipher.
- `int_t cipher_key_length(cipher_t *c)`
Determine the key length of a specified cipher.
- `int_t cipher_vector_length(cipher_t *c)`
Determine the IV length of a specified cipher.

- uint64_t [deprecated_cryptex_body_length](#) (cryptex_t *cryptex)
cryptex.c
- uint64_t [deprecated_cryptex_envelope_length](#) (cryptex_t *cryptex)
Get the length of a cryptex envelope.
- uint64_t [deprecated_cryptex_hmac_length](#) (cryptex_t *cryptex)
Get the length of a cryptex HMAC.
- uint64_t [deprecated_cryptex_original_length](#) (cryptex_t *cryptex)
Get the original length of a cryptex object's data buffer.
- uint64_t [deprecated_cryptex_total_length](#) (cryptex_t *cryptex)
Determine the total length of the data underlying a cryptex object.
- void * [deprecated_cryptex_alloc](#) (uint64_t envelope, uint64_t hmac, uint64_t original, uint64_t body)
Allocate a new cryptex object.
- void * [deprecated_cryptex_body_data](#) (cryptex_t *cryptex)
Get a pointer to the encrypted data body of a cryptex object.
- void * [deprecated_cryptex_envelope_data](#) (cryptex_t *cryptex)
Get a pointer to the envelope data of a cryptex object.
- void * [deprecated_cryptex_hmac_data](#) (cryptex_t *cryptex)
Get a pointer to the HMAC data of a cryptex object.
- void [deprecated_cryptex_free](#) (cryptex_t *cryptex)
Free a cryptex object.
- digest_t * [digest_id](#) (int_t id)
digest.c
- digest_t * [digest_name](#) (stringer_t *name)
- uchr_t * [deprecated_ecies_decrypt](#) (stringer_t *key, ECIES_KEY_TYPE key_type, cryptex_t *cryptex, size_t *length)
ecies.c
- cryptex_t * [deprecated_ecies_encrypt](#) (stringer_t *key, ECIES_KEY_TYPE key_type, unsigned char *data, size_t length)
Encrypt a block of data using an ECIES public key.
- void * [deprecated_ecies_envelope_derivation](#) (const void *input, size_t ilen, void *output, size_t *olen)
- EC_GROUP * [deprecated_ecies_group](#) (uint64_t curve, bool_t precompute)
- EC_KEY * [deprecated_ecies_key_alloc](#) (void)
Allocate a new ECIES key pair from the curve defined by ECIES_CURVE.
- EC_KEY * [deprecated_ecies_key_create](#) (void)
Generate a random ECIES key pair.
- void [deprecated_ecies_key_free](#) (EC_KEY *key)
Free an ECIES key pair.
- EC_KEY * [deprecated_ecies_key_private](#) (uint64_t format, placer_t data)
- uchr_t * [deprecated_ecies_key_private_bin](#) (EC_KEY *key, size_t *olen)

Return an ECIES private key as binary data.

- `stringer_t * deprecated_ecies_key_private_hex` (`EC_KEY *key`)

Return an ECIES private key as a hex string.

- `EC_KEY * deprecated_ecies_key_public` (`uint64_t` format, `placer_t` data)
- `uchr_t * deprecated_ecies_key_public_bin` (`EC_KEY *key`, `size_t *olen`)

Return an ECIES public key as binary data.

- `stringer_t * deprecated_ecies_key_public_hex` (`EC_KEY *key`)

Return an ECIES public key as a null-terminated hex string.

- `bool_t deprecated_ecies_start` (`void`)
- `void deprecated_ecies_stop` (`void`)

Destroy all initialized ECIES curve group information.

- `stringer_t * hash_digest` (`digest_t *digest`, `stringer_t *s`, `stringer_t *output`)

hash.c

- `stringer_t * hash_md4` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * hash_md5` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * hash_ripemd160` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * hash_sha` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * hash_sha1` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * hash_sha224` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * hash_sha256` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * hash_sha384` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * hash_sha512` (`stringer_t *s`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_digest` (`digest_t *digest`, `stringer_t *s`, `stringer_t *key`, `stringer_t *output`)

hmac.c

- `stringer_t * deprecated_hmac_md4` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_md5` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_ripemd160` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_sha` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_sha1` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_sha224` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_sha256` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_sha384` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * deprecated_hmac_sha512` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `bool_t lib_load_openssl` (`void`)

openssl.c

- `const char * lib_version_openssl` (`void`)

Return the version string of the OpenSSL library.

- `DH * ssl_dh2048_exchange_callback` (`SSL *ssl`, `int is_export`, `int keylength`)
- `DH * ssl_dh4096_exchange_callback` (`SSL *ssl`, `int is_export`, `int keylength`)
- `int ssl_dh_generate_callback` (`int p`, `int n`, `BN_GENCB *cb`)
- `EC_KEY * ssl_ecdh_exchange_callback` (`SSL *ssl`, `int is_export`, `int keylength`)

Callback handler for the EC Diffie-Hellman parameter generation process necessary for PFS.

- `char * ssl_error_string` (`chr_t *buffer`, `int_t length`)

Get a textual representation of the last OpenSSL error message.

- void [ssl_locking_callback](#) (int mode, int n, const char *file, int line)
The locking function callback necessary for all multi-threaded applications using OpenSSL.
- [bool_t ssl_start](#) (void)
Initialize the OpenSSL facility.
- void [ssl_stop](#) (void)
Stop the openssl library and cleanup all associated data structures.
- unsigned long [ssl_thread_id_callback](#) (void)
The thread id callback necessary for all multi-threaded applications using OpenSSL.
- void [ssl_thread_stop](#) (void)
Free the calling thread's SSL error queue.
- [bool_t ssl_verify_privkey](#) (const char *keyfile)
Verify that the filename contains a valid private key in PEM format.
- [int_t tls_bits](#) (TLS *tls)
[tls.c](#)
- [stringer_t * tls_cipher](#) (TLS *tls, [stringer_t](#) *output)
Return the name of the TLS cipher suite associated with a connection.
- void * [tls_client_alloc](#) ([int_t](#) sockd)
Establish an TLS client wrapper around a socket descriptor.
- int [tls_continue](#) (TLS *tls, int result, int syserror)
Return -1 if the connection is invalid, 0 if the operation should be retried, or a positive number indicating the number of bytes processed.
- [stringer_t * tls_error](#) (TLS *tls, [int_t](#) code, [stringer_t](#) *output)
Consolidate the complicated logic associated with handling SSL_read/SSL_write calls which result in 0, or a negative number.
- void [tls_free](#) (TLS *tls)
Shutdown and free an TLS connection.
- int [tls_print](#) (TLS *tls, const char *format, va_list args)
Write formatted data to an TLS connection.
- int [tls_read](#) (TLS *tls, void *buffer, int length, [bool_t](#) block)
Read data from a TLS connection.
- TLS * [tls_server_alloc](#) (void *server, int sockd, int flags)
Create a TLS session for a file descriptor, and accept the client TLS/SSL handshake.
- [bool_t tls_server_create](#) (void *server, [uint_t](#) security_level)
Setup an TLS CTX for a server.
- void [tls_server_destroy](#) (void *server)
Destroy an TLS context associated with a server.

- `int tls_status (TLS *tls)`
Checks whether a TLS connection has been shut down or not.
- `chr_t * tls_suite (TLS *tls)`
Provide the SSL/TLS/DTLS cipher suite as a string constant.
- `chr_t * tls_version (TLS *tls)`
Provide the SSL/TLS/DTLS version as a string constant.
- `int tls_write (TLS *tls, const void *buffer, int length, bool_t block)`
Write data to an open TLS connection.
- `bool_t rand_start (void)`
random.c
- `bool_t rand_thread_start (void)`
- `int16_t rand_get_int16 (void)`
- `int32_t rand_get_int32 (void)`
- `int64_t rand_get_int64 (void)`
Generate a random signed 64 bit number.
- `int8_t rand_get_int8 (void)`
- `size_t rand_write (stringer_t *s)`
Fill a managed string with random data.
- `stringer_t * rand_choices (chr_t *choices, size_t len, stringer_t *output)`
Get a random string of data of a specified size, populated with characters from a chosen set.
- `uint16_t rand_get_uint16 (void)`
Generate a random unsigned 16 bit number.
- `uint32_t rand_get_uint32 (void)`
Generate a random unsigned 32 bit number.
- `uint64_t rand_get_uint64 (void)`
Generate a random unsigned 64 bit number.
- `uint8_t rand_get_uint8 (void)`
Generate a random unsigned 8 bit number.
- `void rand_stop (void)`
- `scramble_t * deprecated_scramble_alloc (size_t length)`
scramble.c
- `void * deprecated_scramble_body_data (scramble_t *buffer)`
Get a pointer to the start of the encrypted data from a scrambled data header.
- `uint64_t deprecated_scramble_body_hash (scramble_t *buffer)`
Get a hash of a scrambled object's (encrypted) body data.
- `uint64_t deprecated_scramble_body_length (scramble_t *buffer)`
Get the length of a scrambled object's (encrypted) body data.

- [stringer_t * deprecated_scramble_decrypt](#) ([stringer_t](#) *key, [scramble_t](#) *input)
Un-scramble a block of data using an decryption key.
- [scramble_t * deprecated_scramble_encrypt](#) ([stringer_t](#) *key, [stringer_t](#) *input)
Scramble a block of data using an encryption key.
- void [deprecated_scramble_free](#) ([scramble_t](#) *buffer)
Free a scrambled data block.
- void [deprecated_scramble_cleanup](#) ([scramble_t](#) *buffer)
Performed a checked free of a scramble buffer.
- [scramble_t * deprecated_scramble_import](#) ([stringer_t](#) *s)
Return a managed string as a scrambled buffer, after validation.
- [uint64_t deprecated_scramble_orig_length](#) ([scramble_t](#) *buffer)
Get the length of the original (unencrypted) data underlying a scrambled object.
- [uint64_t deprecated_scramble_total_length](#) ([scramble_t](#) *buffer)
Get the total length of a scrambled object.
- void * [deprecated_scramble_vector_data](#) ([scramble_t](#) *buffer)
Get a pointer to the start of the IV from a scrambled data header.
- [uint64_t deprecated_scramble_vector_length](#) ([scramble_t](#) *buffer)
Get the length of a scrambled object's IV.
- [stringer_t * deprecated_symmetric_decrypt](#) ([cipher_t](#) *cipher, [stringer_t](#) *vector, [stringer_t](#) *key, [stringer_t](#) *input)
symmetric.c
- [stringer_t * deprecated_symmetric_encrypt](#) ([cipher_t](#) *cipher, [stringer_t](#) *vector, [stringer_t](#) *key, [stringer_t](#) *input)
HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.
- [stringer_t * deprecated_symmetric_key](#) ([cipher_t](#) *cipher, [stringer_t](#) *key, [stringer_t](#) *output)
Derive a symmetric key from a user's password.
- [stringer_t * deprecated_symmetric_vector](#) ([cipher_t](#) *cipher, [stringer_t](#) *output)
Generate an IV data suitable for the specified cipher.
- DH * [dh_exchange_2048](#) (SSL *ssl, int is_export, int keylength)
parameters.c
- DH * [dh_exchange_4096](#) (SSL *ssl, int is_export, int keylength)
Callback handler for the Diffie-Hellman parameter generation process necessary for PFS.
- DH * [dh_params_2048](#) (void)
- DH * [dh_params_4096](#) (void)
- void [dh_params_generate](#) (void)
TODO: Spawn a thread at startup to generate the keys, and block the dh_exchange functions below from returning until its done.
- int [dh_params_generate_callback](#) (int p, int n, BN_GENCB *cb)
The DH param generation callback.
- DH * [dh_static_2048](#) (SSL *ssl, int is_export, int keylength)
- DH * [dh_static_4096](#) (SSL *ssl, int is_export, int keylength)

Variables

- [cryptex_head_t](#)
- [scramble_head_t](#)

5.251.1 Define Documentation

5.251.1.1 **#define BN_num_bytes_d(a) ((BN_num_bits_d(a)+7)/8)**

Definition at line 12 of file cryptography.h.

Referenced by deprecated_ecies_key_private_bin(), ecies_key_private_bin(), and secp256k1_private_get().

5.251.1.2 **#define ECIES_CIPHER NID_aes_256_cbc**

Definition at line 24 of file cryptography.h.

Referenced by deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_ecies_start(), ecies_decrypt(), ecies_encrypt(), and ecies_start().

5.251.1.3 **#define ECIES_CURVE NID_sect571k1**

Definition at line 23 of file cryptography.h.

Referenced by deprecated_ecies_key_alloc(), deprecated_ecies_start(), ecies_key_alloc(), and ecies_start().

5.251.1.4 **#define ECIES_ENVELOPE NID_sha512**

Definition at line 25 of file cryptography.h.

Referenced by deprecated_ecies_start(), and ecies_start().

5.251.1.5 **#define ECIES_HMAC NID_sha512**

Definition at line 22 of file cryptography.h.

Referenced by deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_ecies_start(), ecies_decrypt(), ecies_encrypt(), and ecies_start().

5.251.1.6 **#define MAGMA_CIPHERS_GENERIC "HIGH:MEDIUM"**

Definition at line 19 of file cryptography.h.

Referenced by tls_server_create().

5.251.1.7 **#define MAGMA_CIPHERS_HIGH "ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305"**

Definition at line 16 of file cryptography.h.

Referenced by tls_server_create().

```
5.251.1.8 #define MAGMA_CIPHERS_-  
LOW "EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA384:EECDH+ECDSA+SHA256:EECDH  
SHA:!SEED"
```

Definition at line 18 of file cryptography.h.

```
5.251.1.9 #define MAGMA_CIPHERS_-  
MEDIUM "EECDH+ECDSA+AESGCM:EECDH+aRSA+AESGCM:EECDH+ECDSA+SHA384:EECDH+ECDSA+SHA256:!  
SHA:!SEED"
```

Definition at line 17 of file cryptography.h.

Referenced by `tls_server_create()`.

5.251.1.10 #define OPENSSSL free d(addr) CRYPTO free d(addr)

Definition at line 13 of file cryptography.h.

Referenced by deprecated_ecies_key_private_hex(), deprecated_ecies_key_public_hex(), ecies_key_private_hex(), and ecies_key_public_hex().

5.251.2 Typedef Documentation

5.251.2.1 typedef void* cipher_t

Definition at line 62 of file cryptography.h.

5.251.2.2 typedef char* cryptex_t

Definition at line 63 of file cryptography.h.

5.251.2.3 typedef void* digest_t

Definition at line 61 of file cryptography.h.

5.251.2.4 typedef stringer_t scramble_t

Definition at line 64 of file cryptography.h.

5.251.2.5 typedef void TLS

Definition at line 70 of file cryptography.h.

5.251.3 Enumeration Type Documentation

5.251.3.1 enum ECIES_KEY_TYPE

Enumerator:

ECIES PRIVATE HEX

ECIES PRIVATE BINARY

ECIES_PUBLIC_HEX

ECIES_PUBLIC_BINARY

Definition at line 27 of file cryptography.h.

5.251.4 Function Documentation

5.251.4.1 `struct __attribute__((packed)) [read]`

Definition at line 44 of file cryptography.h.

References `length`.

5.251.4.2 `int_t cipher_block_length (cipher_t * c)`

Determine the block length of a specified cipher.

Parameters:

c the input cipher type.

Returns:

-1 on failure, or the cipher's block length in bytes.

Definition at line 94 of file ciphers.c.

References `EVP_CIPHER_block_size_d`, `EVP_CIPHER_nid_d`, `log_pedantic`, and `OBJ_nid2sn_d`.

5.251.4.3 `cipher_t* cipher_id (int_t id)`

[ciphers.c](#) [ciphers.c](#)

Parameters:

id the numerical ID (nid) of the specified cipher.

Returns:

NULL on failure, or the relevant cipher data structure on success.

Definition at line 30 of file ciphers.c.

References `EVP_get_cipherbyname_d`, `log_pedantic`, and `OBJ_nid2sn_d`.

Referenced by `deprecated_scramble_decrypt()`, `deprecated_scramble_encrypt()`, `scramble_decrypt()`, and `scramble_encrypt()`.

5.251.4.4 `int_t cipher_key_length (cipher_t * c)`

Determine the key length of a specified cipher.

Parameters:

c the input cipher type.

Returns:

-1 on failure, or the cipher's key length in bytes.

Definition at line 62 of file ciphers.c.

References `EVP_CIPHER_key_length_d`, `EVP_CIPHER_nid_d`, `log_pedantic`, and `OBJ_nid2sn_d`.

Referenced by `deprecated_symmetric_key()`, and `symmetric_key()`.

5.251.4.5 `cipher_t* cipher_name (stringer_t * name)`

Look up a cipher type by name.

Parameters:

name a descriptive name of the cipher to be looked up.

Returns:

NULL on failure, or the relevant cipher data structure on success.

Definition at line 15 of file ciphers.c.

References `EVP_get_cipherbyname_d`, `log_pedantic`, `st_char_get()`, `st_empty`, and `st_length_int()`.

5.251.4.6 `int_t cipher_numeric_id (cipher_t * c)`

Return the numerical ID (nid) of a specified cipher.

Parameters:

c the input cipher type.

Returns:

`NID_undef` on failure, or the nid of the specified cipher.

Definition at line 46 of file ciphers.c.

References `EVP_CIPHER_nid_d`, and `log_pedantic`.

Referenced by `deprecated_scramble_encrypt()`, and `scramble_encrypt()`.

5.251.4.7 `int_t cipher_vector_length (cipher_t * c)`

Determine the IV length of a specified cipher.

Parameters:

c the input cipher type.

Returns:

-1 on failure, or the cipher's IV length in bytes.

Definition at line 78 of file ciphers.c.

References `EVP_CIPHER_iv_length_d`, `EVP_CIPHER_nid_d`, `log_pedantic`, and `OBJ_nid2sn_d`.

Referenced by `deprecated_symmetric_vector()`, and `symmetric_vector()`.

5.251.4.8 void* deprecated_cryptex_alloc (uint64_t *envelope*, uint64_t *hmac*, uint64_t *original*, uint64_t *body*)

Allocate a new cryptex object.

Definition at line 108 of file cryptex.c.

References cryptex_head_t, log_pedantic, and mm_alloc().

Referenced by deprecated_ecies_encrypt().

5.251.4.9 void* deprecated_cryptex_body_data (cryptex_t * *cryptex*)

Get a pointer to the encrypted data body of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's encrypted data body.

Definition at line 97 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.251.4.10 uint64_t deprecated_cryptex_body_length (cryptex_t * *cryptex*)

cryptex.c cryptex.c

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex encrypted data body.

Definition at line 39 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.251.4.11 void* deprecated_cryptex_envelope_data (cryptex_t * *cryptex*)

Get a pointer to the envelope data of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's envelope data.

Definition at line 75 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.251.4.12 uint64_t deprecated_cryptex_envelope_length (cryptex_t * cryptex)

Get the length of a cryptex envelope.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex envelope.

Definition at line 15 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt().

5.251.4.13 void deprecated_cryptex_free (cryptex_t * cryptex)

Free a cryptex object.

Parameters:

cryptex the cryptex object to be freed.

Returns:

This function returns no value.

Definition at line 132 of file cryptex.c.

Referenced by deprecated_ecies_encrypt().

5.251.4.14 void* deprecated_cryptex_hmac_data (cryptex_t * cryptex)

Get a pointer to the HMAC data of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's hmac data.

Definition at line 85 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.251.4.15 uint64_t deprecated_cryptex_hmac_length (cryptex_t * cryptex)

Get the length of a cryptex HMAC.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex HMAC.

Definition at line 27 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.251.4.16 uint64_t deprecated_cryptex_original_length (cryptex_t * cryptex)

Get the original length of a cryptex object's data buffer.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex object's original data.

Definition at line 51 of file cryptex.c.

References cryptex_head_t.

Referenced by deprecated_ecies_decrypt().

5.251.4.17 uint64_t deprecated_cryptex_total_length (cryptex_t * cryptex)

Determine the total length of the data underlying a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

the total length of the associated cryptex data in bytes.

Definition at line 63 of file cryptex.c.

References cryptex_head_t.

5.251.4.18 uchr_t* deprecated_ecies_decrypt (stringer_t * key, ECIES_KEY_TYPE key_type, cryptex_t * cryptex, size_t * length)

ecies.c ecies.c

Parameters:

key the ECIES private key in the specified format.

key_type the encoding type of the ECIES private key (ECIES_PRIVATE_BINARY or ECIES_PRIVATE_HEX).

cryptex a pointer to the head of the cryptex object with the encrypted data.

length a pointer to a size_t variable which will receive the final length of the unencrypted data.

Returns:

NULL on failure, or a pointer to a memory address containing the decrypted data on success..

Definition at line 578 of file ecies.c.

References deprecated_cryptex_body_data(), deprecated_cryptex_body_length(), deprecated_cryptex_envelope_data(), deprecated_cryptex_envelope_length(), deprecated_cryptex_hmac_data(), deprecated_cryptex_hmac_length(), deprecated_cryptex_original_length(), deprecated_ecies_envelope_derivation(), deprecated_ecies_key_private(), deprecated_ecies_key_public(), EC_KEY_free_d, EC_KEY_get0_public_key_d, ECDH_compute_key_d, ECIES_CIPHER, ECIES_HMAC, ECIES_PRIVATE_BINARY, ECIES_PRIVATE_HEX, ECIES_PUBLIC_BINARY, ECIES_PUBLIC_HEX, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_set_padding_d, EVP_CIPHER_key_length_d, EVP_DecryptFinal_ex_d, EVP_DecryptInit_ex_d, EVP_DecryptUpdate_d, EVP_get_cipherbyname_d, EVP_get_digestbyname_d, HMAC_CTX_cleanup_d, HMAC_CTX_init_d, HMAC_Final_d, HMAC_Init_ex_d, HMAC_Update_d, log_info, MEMORYBUF, mm_alloc(), OBJ_nid2sn_d, pl_init(), ssl_error_string(), and st_empty_out().

5.251.4.19 cryptex_t* deprecated_ecies_encrypt (stringer_t *key, ECIES_KEY_TYPE key_type, unsigned char *data, size_t length)

Encrypt a block of data using an ECIES public key.

Parameters:

key the ECIES public key in the specified format.

key_type the encoding type of the ECIES public key (ECIES_PUBLIC_BINARY or ECIES_PUBLIC_HEX).

data a pointer to the block of data to be encrypted.

length the length, in bytes, of the data to be encrypted.

Returns:

NULL on failure, or a pointer to the header of the cryptex object containing the encrypted data on success..

Definition at line 407 of file ecies.c.

References deprecated_cryptex_alloc(), deprecated_cryptex_body_data(), deprecated_cryptex_body_length(), deprecated_cryptex_envelope_data(), deprecated_cryptex_free(), deprecated_cryptex_hmac_data(), deprecated_cryptex_hmac_length(), deprecated_ecies_envelope_derivation(), deprecated_ecies_key_create(), deprecated_ecies_key_public(), EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2oct_d, ECDH_compute_key_d, ECIES_CIPHER, ECIES_HMAC, ECIES_PUBLIC_BINARY, ECIES_PUBLIC_HEX, EVP_CIPHER_block_size_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_set_padding_d, EVP_CIPHER_key_length_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, EVP_get_cipherbyname_d, EVP_get_digestbyname_d, EVP_MD_size_d, HMAC_CTX_cleanup_d, HMAC_CTX_init_d, HMAC_Final_d, HMAC_Init_ex_d, HMAC_Update_d, log_info, MEMORYBUF, OBJ_nid2sn_d, pl_init(), ssl_error_string(), and st_empty_out().

5.251.4.20 void* deprecated_ecies_envelope_derivation (const void *input, size_t ilen, void *output, size_t olen)

Definition at line 390 of file ecies.c.

References ecies_envelope_evpc, and EVP_Digest_d.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.251.4.21 EC_GROUP* deprecated_ecies_group (uint64_t curve, bool_t precompute)

Definition at line 89 of file ecies.c.

References EC_GROUP_free_d, EC_GROUP_new_by_curve_name_d, EC_GROUP_precompute_mult_d, EC_GROUP_set_point_conversion_form_d, log_error, MEMORYBUF, and ssl_error_string().

Referenced by deprecated_ecies_start().

5.251.4.22 EC_KEY* deprecated_ecies_key_alloc (void)

Allocate a new ECIES key pair from the curve defined by ECIES_CURVE.

See also:

NID_sect571k1

Returns:

NULL on failure, or a pointer to the newly allocated key pair.

Definition at line 58 of file ecies.c.

References EC_KEY_free_d, EC_KEY_new_by_curve_name_d, EC_KEY_new_d, EC_KEY_set_conv_form_d, EC_KEY_set_group_d, ECIES_CURVE, ecies_curve_group, log_info, MEMORYBUF, and ssl_error_string().

Referenced by deprecated_ecies_key_create(), deprecated_ecies_key_private(), and deprecated_ecies_key_public().

5.251.4.23 EC_KEY* deprecated_ecies_key_create (void)

Generate a random ECIES key pair.

Returns:

NULL on failure, or a new random ECIES key pair on success.

Definition at line 112 of file ecies.c.

References deprecated_ecies_key_alloc(), EC_KEY_free_d, EC_KEY_generate_key_d, log_info, MEMORYBUF, and ssl_error_string().

Referenced by deprecated_ecies_encrypt().

5.251.4.24 void deprecated_ecies_key_free (EC_KEY * key)

Free an ECIES key pair.

Returns:

This function returns no value.

Definition at line 47 of file ecies.c.

References EC_KEY_free_d.

5.251.4.25 EC_KEY* deprecated_ecies_key_private (uint64_t format, placer_t data)

Definition at line 199 of file ecies.c.

References BN_bin2bn_d, BN_free_d, BN_hex2bn_d, deprecated_ecies_key_alloc(), EC_KEY_free_d, EC_KEY_set_private_key_d, ECIES_PRIVATE_BINARY, ECIES_PRIVATE_HEX, log_info, MEMORYBUF, number, pl_char_get(), pl_data_get(), pl_length_get(), and ssl_error_string().

Referenced by deprecated_ecies_decrypt().

5.251.4.26 uchr_t* deprecated_ecies_key_private_bin (EC_KEY * key, size_t * olen)

Return an ECIES private key as binary data.

Parameters:

key the input ECIES key pair.

olen a pointer to store the length of the returned key.

Returns:

NULL on failure, or a pointer to the raw private key.

Definition at line 359 of file ecies.c.

References BN_bn2bin_d, BN_num_bytes_d, EC_KEY_get0_private_key_d, log_info, MEMORYBUF, mm_sec_alloc(), mm_sec_free(), and ssl_error_string().

5.251.4.27 stringer_t* deprecated_ecies_key_private_hex (EC_KEY * *key*)

Return an ECIES private key as a hex string.

Parameters:

key the input ECIES key pair.

Returns:

NULL on failure, or the hex-formatted private key as a managed string.

Definition at line 325 of file ecies.c.

References BN_bn2hex_d, CONTIGUOUS, EC_KEY_get0_private_key_d, log_pedantic, MANAGED_T, MEMORYBUF, ns_length_get(), ns_wipe(), OPENSSL_free_d, SECURE, ssl_error_string(), and st_import_opts().

5.251.4.28 EC_KEY* deprecated_ecies_key_public (uint64_t *format*, placer_t *data*)

Definition at line 131 of file ecies.c.

References deprecated_ecies_key_alloc(), EC_KEY_check_key_d, EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_set_public_key_d, EC_POINT_free_d, EC_POINT_hex2point_d, EC_POINT_new_d, EC_POINT_oct2point_d, ECIES_PUBLIC_BINARY, ECIES_PUBLIC_HEX, log_info, MEMORYBUF, pl_char_get(), pl_data_get(), pl_length_get(), and ssl_error_string().

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.251.4.29 uchr_t* deprecated_ecies_key_public_bin (EC_KEY * *key*, size_t * *olen*)

Return an ECIES public key as binary data.

Parameters:

key the input ECIES key pair.

olen a pointer to store the length of the returned key.

Returns:

NULL on failure, or a pointer to the raw public key.

Definition at line 288 of file ecies.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2oct_d, log_info, MEMORYBUF, mm_alloc(), mm_free(), and ssl_error_string().

5.251.4.30 stringer_t* deprecated_ecies_key_public_hex (EC_KEY * *key*)

Return an ECIES public key as a null-terminated hex string.

Parameters:

key the input ECIES key pair.

Returns:

NULL on failure, or the hex-formatted public key as a null-terminated string.

Definition at line 255 of file ecies.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2hex_d, log_info, MEMORYBUF, ns_length_get(), OPENSSL_free_d, ssl_error_string(), and st_import().

5.251.4.31 bool_t deprecated_ecies_start (void)

Definition at line 15 of file ecies.c.

References deprecated_ecies_group(), ECIES_CIPHER, ecies_cipher_evp, ECIES_CURVE, ecies_curve_group, ECIES_ENVELOPE, ecies_envelope_evp, ECIES_HMAC, ecies_hmac_evp, EVP_get_cipherbyname_d, EVP_get_digestbyname_d, log_error, and OBJ_nid2sn_d.

Referenced by process_start().

5.251.4.32 void deprecated_ecies_stop (void)

Destroy all initialized ECIES curve group information.

Returns:

This function returns no value.

Definition at line 30 of file ecies.c.

References EC_GROUP_free_d, and ecies_curve_group.

Referenced by process_stop().

5.251.4.33 stringer_t* deprecated_hmac_digest (digest_t * digest, stringer_t * s, stringer_t * key, stringer_t * output)

hmac.c hmac.c

Parameters:

digest Digest to be used with the HMAC.

s Input data.

key Key used in HMAC.

output Stringer containing buffer for output.

Returns:

The managed string containing the resulting HMAC or NULL if an error occurs.

Definition at line 20 of file hmac.c.

References EVP_MD_size_d, HMAC_CTX_cleanup_d, HMAC_CTX_init_d, HMAC_Final_d, HMAC_Init_ex_d, HMAC_Update_d, log_pedantic, st_alloc(), st_avail_get(), st_data_get(), st_empty, st_free(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), and st_valid_tracked().

Referenced by deprecated_hmac_md4(), deprecated_hmac_md5(), deprecated_hmac_ripemd160(), deprecated_hmac_sha(), deprecated_hmac_sha1(), deprecated_hmac_sha224(), deprecated_hmac_sha256(), deprecated_hmac_sha384(), and deprecated_hmac_sha512().

5.251.4.34 stringer_t* deprecated_hmac_md4 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 90 of file hmac.c.

References deprecated_hmac_digest(), and EVP_md4_d.

5.251.4.35 stringer_t* deprecated_hmac_md5 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 94 of file hmac.c.

References deprecated_hmac_digest(), and EVP_md5_d.

5.251.4.36 stringer_t* deprecated_hmac_ripemd160 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 122 of file hmac.c.

References deprecated_hmac_digest(), and EVP_ripemd160_d.

5.251.4.37 stringer_t* deprecated_hmac_sha (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 98 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha_d.

5.251.4.38 stringer_t* deprecated_hmac_sha1 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 102 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha1_d.

5.251.4.39 stringer_t* deprecated_hmac_sha224 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 106 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha224_d.

5.251.4.40 stringer_t* deprecated_hmac_sha256 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 110 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha256_d.

5.251.4.41 stringer_t* deprecated_hmac_sha384 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 114 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha384_d.

5.251.4.42 stringer_t* deprecated_hmac_sha512 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 118 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha512_d.

5.251.4.43 `scramble_t* deprecated_scramble_alloc (size_t length)`

scramble.c scramble.c

Parameters:

length the length, in bytes, of the scrambled data buffer (should include the IV and encrypted body length).

Returns:

a pointer to a newly allocated scrambled data header.

Definition at line 143 of file scramble.c.

References `mm_alloc()`, and `scramble_head_t`.

Referenced by `deprecated_scramble_encrypt()`.

5.251.4.44 `void* deprecated_scramble_body_data (scramble_t * buffer)`

Get a pointer to the start of the encrypted data from a scrambled data header.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

a pointer to the start of the scrambled data header's associated encrypted data body.

Definition at line 81 of file scramble.c.

References `deprecated_scramble_vector_length()`, and `scramble_head_t`.

Referenced by `deprecated_scramble_decrypt()`, `deprecated_scramble_encrypt()`, and `deprecated_scramble_import()`.

5.251.4.45 `uint64_t deprecated_scramble_body_hash (scramble_t * buffer)`

Get a hash of a scrambled object's (encrypted) body data.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the 64-bit hash of the specified scrambled object's body data.

Definition at line 33 of file scramble.c.

References `scramble_head_t`.

5.251.4.46 `uint64_t deprecated_scramble_body_length (scramble_t * buffer)`

Get the length of a scrambled object's (encrypted) body data.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's body data.

Definition at line 57 of file scramble.c.

References `scramble_head_t`.

Referenced by `deprecated_scramble_decrypt()`, and `deprecated_scramble_total_length()`.

5.251.4.47 void deprecated_scramble_cleanup (scramble_t * *buffer*)

Performed a checked free of a scramble buffer.

See also:

[scramble_free](#)

Parameters:

block the scramble buffer to be freed.

Returns:

This function returns no value.

Definition at line 166 of file scramble.c.

References `mm_free()`.

5.251.4.48 stringer_t* deprecated_scramble_decrypt (stringer_t * *key*, scramble_t * *input*)

Un-scramble a block of data using an decryption key.

Parameters:

key a managed string containing the symmetric decryption key.

input a pointer to the scrambled data header to be decrypted.

Returns:

NULL on failure, or a managed string containing the verified decrypted data.

Definition at line 239 of file scramble.c.

References `cipher_id()`, `deprecated_scramble_body_data()`, `deprecated_scramble_body_length()`, `deprecated_scramble_vector_data()`, `deprecated_scramble_vector_length()`, `deprecated_symmetric_decrypt()`, `deprecated_symmetric_key()`, `hash_adler32()`, `log_info`, `log_pedantic`, `MANAGEDBUF`, `PLACER`, `scramble_head_t`, `st_free()`, and `st_length_get()`.

Referenced by `sess_get()`.

5.251.4.49 scramble_t* deprecated_scramble_encrypt (stringer_t * *key*, stringer_t * *input*)

Scramble a block of data using an encryption key.

Note:

This function is configured to use AES 256 in CBC mode.

Parameters:

key a managed string containing the symmetric encryption key.

input a managed string containing the data to be encrypted.

Returns:

NULL on failure, or a pointer to a scrambled data header containing the encrypted data and its metadata.

Definition at line 182 of file scramble.c.

References cipher_id(), cipher_numeric_id(), deprecated_scramble_alloc(), deprecated_scramble_body_data(), deprecated_scramble_vector_data(), deprecated_symmetric_encrypt(), deprecated_symmetric_key(), deprecated_symmetric_vector(), hash_adler32(), log_pedantic, MANAGEDBUF, mm_copy(), scramble_head_t, st_data_get(), st_free(), and st_length_get().

Referenced by sess_token().

5.251.4.50 void deprecated_scramble_free (scramble_t * *buffer*)

Free a scrambled data block.

Parameters:

buffer a pointer to the scrambled data header to be freed.

Returns:

This function returns no value.

Definition at line 153 of file scramble.c.

References mm_free().

Referenced by sess_token().

5.251.4.51 scramble_t* deprecated_scramble_import (stringer_t * *s*)

Return a managed string as a scrambled buffer, after validation.

Note:

The returned pointer is inside the existing stringer, so it doesn't need be freed.

Parameters:

s a managed string containing the serialized scrambled data.

Returns:

NULL on failure, or a pointer to the scrambled data header on success.

Definition at line 102 of file scramble.c.

References deprecated_scramble_body_data(), hash_adler32(), log_pedantic, scramble_head_t, st_data_get(), st_empty, and st_length_get().

Referenced by sess_get().

5.251.4.52 uint64_t deprecated_scramble_orig_length (scramble_t * *buffer*)

Get the length of the original (unencrypted) data underlying a scrambled object.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's original data.

Definition at line 45 of file scramble.c.

References `scramble_head_t`.

5.251.4.53 uint64_t deprecated_scramble_total_length (scramble_t * *buffer*)

Get the total length of a scrambled object. TODO: Create a `scramble_export()` function, and/or alter the import interfaces to make using managed strings easier. ie. the import/export functions could provide wrappers around `dseccrypt/encrypt` which return strings instead of scramble objects. HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the total length, in bytes, of the scrambled object.

Definition at line 19 of file scramble.c.

References `deprecated_scramble_body_length()`, `deprecated_scramble_vector_length()`, and `scramble_head_t`.

Referenced by `sess_token()`.

5.251.4.54 void* deprecated_scramble_vector_data (scramble_t * *buffer*)

Get a pointer to the start of the IV from a scrambled data header.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

a pointer to the start of the scrambled data header's associated IV block.

Definition at line 91 of file scramble.c.

References `scramble_head_t`.

Referenced by `deprecated_scramble_decrypt()`, and `deprecated_scramble_encrypt()`.

5.251.4.55 uint64_t deprecated_scramble_vector_length (scramble_t * *buffer*)

Get the length of a scrambled object's IV.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's IV.

Definition at line 69 of file scramble.c.

References `scramble_head_t`.

Referenced by `deprecated_scramble_body_data()`, `deprecated_scramble_decrypt()`, and `deprecated_scramble_total_length()`.

5.251.4.56 **stringer_t* deprecated_symmetric_decrypt** (cipher_t * *cipher*, stringer_t * *vector*, stringer_t * *key*, stringer_t * *input*)

symmetric.c

Definition at line 128 of file symmetric.c.

References EVP_CIPHER_CTX_block_size_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_iv_length_d, EVP_CIPHER_CTX_key_length_d, EVP_CIPHER_flags_d, EVP_DecryptFinal_ex_d, EVP_DecryptInit_ex_d, EVP_DecryptUpdate_d, log_pedantic, MEMORYBUF, ssl_error_string(), st_alloc(), st_data_get(), st_empty_out(), st_free(), and st_length_set().

Referenced by deprecated_scramble_decrypt().

5.251.4.57 **stringer_t* deprecated_symmetric_encrypt** (cipher_t * *cipher*, stringer_t * *vector*, stringer_t * *key*, stringer_t * *input*)

HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory. Encrypt a block of data using a symmetric cipher.

Parameters:

- cipher*** the cipher type being used to encrypt the data.
- vector*** the IV used for the encryption operation.
- key*** the symmetric encryption key used to encrypt the data.
- input*** a managed string containing the data to be encrypted.

Returns:

NULL on failure, or a pointer to a managed string containing the encrypted data.

Definition at line 20 of file symmetric.c.

References EVP_CIPHER_CTX_block_size_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_iv_length_d, EVP_CIPHER_CTX_key_length_d, EVP_CIPHER_flags_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, log_pedantic, st_alloc(), st_data_get(), st_empty_out(), st_free(), and st_length_set().

Referenced by deprecated_scramble_encrypt().

5.251.4.58 **stringer_t* deprecated_symmetric_key** (cipher_t * *cipher*, stringer_t * *key*, stringer_t * *output*)

Derive a symmetric key from a user's password.

Note:

Depending on the length of the password, either SHA1, SHA224, or SHA256 may be used as the hashing algorithm. If output is passed as NULL, a new managed string will be allocated to hold the output of the operation.

Parameters:

- cipher*** the specified cipher type.
- key*** the user's password, as a managed string.
- output*** the output managed string to receive the digested password. Can be NULL.

Returns:

NULL on failure, or a pointer to the managed string containing the key.

Definition at line 288 of file symmetric.c.

References cipher_key_length(), hash_sha1(), hash_sha224(), hash_sha256(), st_length_get(), st_length_set(), st_output(), and st_valid_tracked().

Referenced by deprecated_scramble_decrypt(), and deprecated_scramble_encrypt().

5.251.4.59 stringer_t* deprecated_symmetric_vector (cipher_t * cipher, stringer_t * output)

Generate an IV data suitable for the specified cipher.

Note:

If output is NULL, a new managed string will be allocated and returned by the function.

Parameters:

cipher the cipher type to be used.

output the managed string that will store the IV data.

Returns:

NULL on failure, or a pointer to the managed string that contains the IV data.

Definition at line 251 of file symmetric.c.

References cipher_vector_length(), log_pedantic, PLACER, rand_write(), st_data_get(), st_free(), st_length_set(), st_output(), and st_valid_tracked().

Referenced by deprecated_scramble_encrypt().

5.251.4.60 DH* dh_exchange_2048 (SSL * ssl, int is_export, int keylength)

[parameters.c](#) [parameters.c](#)

Parameters:

ssl the SSL session for which the callback was triggered.

is_export We ignore this parameter, as we don't support insecure cipher suites.

keylength the length, in bits, of the DH key to be generated is also ignored.

Returns:

a pointer to a Diffie-Hellman structure with a 2048 bit prime which is used for the session key, or NULL on failure.

Definition at line 270 of file parameters.c.

References dh2048, DH_check_d, DH_free_d, DH_generate_parameters_ex_d, DH_new_d, dh_params_generate_callback(), dhparam_lock, log_error, log_pedantic, MEMORYBUF, mutex_lock(), mutex_unlock(), ssl_error_string(), and status.

Referenced by tls_server_create().

5.251.4.61 DH* dh_exchange_4096 (SSL * ssl, int is_export, int keylength)

Callback handler for the Diffie-Hellman parameter generation process necessary for PFS.

Parameters:

ssl the SSL session for which the callback was triggered.

is_export We ignore this parameter, as we don't support insecure cipher suites.

keylength the length, in bits, of the DH key to be generated is also ignored.

Returns:

a pointer to a Diffie-Hellman structure with a 4096 bit prime which is used for the session key, or NULL on failure.

Definition at line 338 of file parameters.c.

References dh4096, DH_check_d, DH_free_d, DH_generate_parameters_ex_d, DH_new_d, dh_params_generate_callback(), dhparam_lock, log_error, log_pedantic, MEMORYBUF, mutex_lock(), mutex_unlock(), ssl_error_string(), and status.

Referenced by tls_server_create().

5.251.4.62 DH* dh_params_2048 (void)

Definition at line 140 of file parameters.c.

References BN_bin2bn_d, DH_free_d, and DH_new_d.

Referenced by dh_params_generate(), and dh_static_2048().

5.251.4.63 DH* dh_params_4096 (void)

Definition at line 187 of file parameters.c.

References BN_bin2bn_d, DH_free_d, and DH_new_d.

Referenced by dh_params_generate(), and dh_static_4096().

5.251.4.64 void dh_params_generate (void)

TODO: Spawn a thread at startup to generate the keys, and block the dh_exchange functions below from returning until its done.

Definition at line 25 of file parameters.c.

References magma_t::cryptography, dh2048, dh4096, DH_check_d, DH_free_d, DH_generate_parameters_ex_d, DH_new_d, dh_params_2048(), dh_params_4096(), dh_params_generate_callback(), dhparam_lock, magma_t::iface, log_error, log_pedantic, magma, MEMORYBUF, mutex_lock(), mutex_unlock(), ssl_error_string(), and status.

5.251.4.65 int dh_params_generate_callback (int p, int n, BN_GENCB * cb)

The DH param generation callback.

Definition at line 16 of file parameters.c.

References status.

Referenced by dh_exchange_2048(), dh_exchange_4096(), and dh_params_generate().

5.251.4.66 DH* dh_static_2048 (SSL * ssl, int is_export, int keylength)

Definition at line 255 of file parameters.c.

References dh_params_2048().

Referenced by tls_server_create().

5.251.4.67 DH* dh_static_4096 (SSL * ssl, int is_export, int keylength)

Definition at line 259 of file parameters.c.

References dh_params_4096().

Referenced by tls_server_create().

5.251.4.68 `digest_t* digest_id (int_t id)`[digest.c](#)

Definition at line 21 of file digest.c.

References `EVP_get_digestbyname_d`, `log_pedantic`, `MEMORYBUF`, `OBJ_nid2sn_d`, and `ssl_error_string()`.

5.251.4.69 `digest_t* digest_name (stringer_t * name)`

Definition at line 10 of file digest.c.

References `EVP_get_digestbyname_d`, `log_pedantic`, `MEMORYBUF`, `ssl_error_string()`, `st_char_get()`, `st_empty`, and `st_length_int()`.

5.251.4.70 `stringer_t* hash_digest (digest_t * digest, stringer_t * s, stringer_t * output)`[hash.c](#)

Definition at line 10 of file hash.c.

References `EVP_DigestFinal_d`, `EVP_DigestInit_ex_d`, `EVP_DigestUpdate_d`, `EVP_MD_CTX_cleanup_d`, `EVP_MD_CTX_init_d`, `EVP_MD_size_d`, `log_pedantic`, `MEMORYBUF`, `ssl_error_string()`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `hash_md4()`, `hash_md5()`, `hash_ripemd160()`, `hash_sha()`, `hash_sha1()`, `hash_sha224()`, `hash_sha256()`, `hash_sha384()`, and `hash_sha512()`.

5.251.4.71 `stringer_t* hash_md4 (stringer_t * s, stringer_t * output)`

Definition at line 86 of file hash.c.

References `EVP_md4_d`, and `hash_digest()`.

5.251.4.72 `stringer_t* hash_md5 (stringer_t * s, stringer_t * output)`

Definition at line 90 of file hash.c.

References `EVP_md5_d`, and `hash_digest()`.

5.251.4.73 `stringer_t* hash_ripemd160 (stringer_t * s, stringer_t * output)`

Definition at line 118 of file hash.c.

References `EVP_ripemd160_d`, and `hash_digest()`.

5.251.4.74 `stringer_t* hash_sha (stringer_t * s, stringer_t * output)`

Definition at line 94 of file hash.c.

References `EVP_sha_d`, and `hash_digest()`.

5.251.4.75 `stringer_t* hash_sha1 (stringer_t * s, stringer_t * output)`

Definition at line 98 of file hash.c.

References `EVP_sha1_d`, and `hash_digest()`.

Referenced by `deprecated_symmetric_key()`, and `symmetric_key()`.

5.251.4.76 stringer_t* hash_sha224 (stringer_t * s, stringer_t * output)

Definition at line 102 of file hash.c.

References `EVP_sha224_d`, and `hash_digest()`.

Referenced by `deprecated_symmetric_key()`, and `symmetric_key()`.

5.251.4.77 stringer_t* hash_sha256 (stringer_t * s, stringer_t * output)

Definition at line 106 of file hash.c.

References `EVP_sha256_d`, and `hash_digest()`.

Referenced by `deprecated_symmetric_key()`, and `symmetric_key()`.

5.251.4.78 stringer_t* hash_sha384 (stringer_t * s, stringer_t * output)

Definition at line 110 of file hash.c.

References `EVP_sha384_d`, and `hash_digest()`.

5.251.4.79 stringer_t* hash_sha512 (stringer_t * s, stringer_t * output)

Definition at line 114 of file hash.c.

References `EVP_sha512_d`, and `hash_digest()`.

Referenced by `auth_legacy()`, `encrypted_chunk_get()`, `encrypted_chunk_set()`, `org_signet_fingerprint()`, `signature_full_get()`, `signature_full_verify()`, `signature_tree_add()`, `signature_tree_get()`, `signature_tree_verify()`, and `user_signet_fingerprint()`.

5.251.4.80 bool_t lib_load_openssl (void)

[openssl.c](#) [openssl.c](#)

Returns:

true on success or false on failure.

Definition at line 29 of file openssl.c.

References `lib_symbols()`, `M_BIND`, `ns_length_get()`, `pl_char_get()`, `pl_length_int()`, `placer_t`, `ssl_version`, `SSL_version_str_d`, `symbol_t`, and `tok_get_ns()`.

Referenced by `lib_load()`.

5.251.4.81 const char* lib_version_openssl (void)

Return the version string of the OpenSSL library.

Returns:

a pointer to a character string containing the libopenssl version information.

Definition at line 20 of file openssl.c.

References `ssl_version`.

Referenced by `lib_load()`.

5.251.4.82 stringer_t* rand_choices (chr_t * *choices*, size_t *len*, stringer_t * *output*)

Get a random string of data of a specified size, populated with characters from a chosen set.

Parameters:

choices a pointer to a null-terminated string containing a pool of characters from which the contents of the random data will be selected.
len the length, in bytes, of the random string that will be returned to the caller.

Returns:

NULL on failure or a pointer to a managed string containing len bytes of random data on success.

Definition at line 23 of file random.c.

References log_pedantic, ns_length_get(), RAND_bytes_d, st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), and st_valid_tracked().

Referenced by mail_create_message(), register_print_captcha(), register_session_generate(), smtp_bounce(), and smtp_reply().

5.251.4.83 int16_t rand_get_int16 (void)

Definition at line 248 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

5.251.4.84 int32_t rand_get_int32 (void)

Definition at line 227 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

5.251.4.85 int64_t rand_get_int64 (void)

Generate a random signed 64 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated signed 64 bit integer.

See also:

RAND_bytes()

Definition at line 204 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

Referenced by portal_message_source().

5.251.4.86 int8_t rand_get_int8 (void)

Definition at line 269 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

5.251.4.87 uint16_t rand_get_uint16 (void)

Generate a random unsigned 16 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated unsigned 16 bit integer.

See also:

RAND_bytes()

Definition at line 159 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

5.251.4.88 uint32_t rand_get_uint32 (void)

Generate a random unsigned 32 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated unsigned 32 bit integer.

See also:

RAND_bytes()

Definition at line 137 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

Referenced by auth_login(), process_maint(), register_captcha_generate(), register_captcha_random_font(), register_captcha_write_noise(), and smtp_client_connect().

5.251.4.89 uint64_t rand_get_uint64 (void)

Generate a random unsigned 64 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated unsigned 64 bit integer.

See also:

RAND_bytes()

Definition at line 113 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

Referenced by mail_mime_generate_boundary(), sess_key(), smtp_store_spamsig(), and spool_mktemp().

5.251.4.90 uint8_t rand_get_uint8 (void)

Generate a random unsigned 8 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated unsigned 8 bit integer.

See also:

RAND_bytes()

Definition at line 181 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

5.251.4.91 bool_t rand_start (void)

[random.c](#) [random.c](#)

Note:

The default seed source for cryptographically secure generation routines is the system device /dev/random.

Returns:

false on failure, true on success.

Definition at line 308 of file random.c.

References magma_t::cryptography, magma_t::iface, magma, rand_ctx, RAND_load_file_d, RAND_status_d, and thread_get_thread_id().

Referenced by process_start().

5.251.4.92 void rand_stop (void)

Definition at line 331 of file random.c.

References RAND_cleanup_d.

Referenced by process_stop().

5.251.4.93 bool_t rand_thread_start (void)

Definition at line 292 of file random.c.

References magma_t::cryptography, magma_t::iface, magma, rand_ctx, and thread_get_thread_id().

5.251.4.94 size_t rand_write (stringer_t * s)

Fill a managed string with random data.

Note:

This function generates random data using the cryptographically strong OpenSSL function RAND_bytes().

Parameters:

s the input managed string.

Returns:

0 on failure, or the total number of bytes written to the managed string.

See also:

RAND_bytes()

Definition at line 76 of file random.c.

References log_pedantic, RAND_bytes_d, st_avail_get(), st_data_get(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), and st_valid_tracked().

Referenced by aes_artifact_encrypt(), aes_chunk_encrypt(), deprecated_symmetric_vector(), encrypted_chunk_set(), signature_full_get(), signature_tree_get(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_encrypt(), and symmetric_vector().

5.251.4.95 DH* ssl_dh2048_exchange_callback (SSL **ssl*, int *is_export*, int *keylength*)

5.251.4.96 DH* ssl_dh4096_exchange_callback (SSL **ssl*, int *is_export*, int *keylength*)

5.251.4.97 int ssl_dh_generate_callback (int *p*, int *n*, BN_GENCB **cb*)

5.251.4.98 EC_KEY* ssl_ecdh_exchange_callback (SSL **ssl*, int *is_export*, int *keylength*)

Callback handler for the EC Diffie-Hellman parameter generation process necessary for PFS.

Parameters:

ssl the SSL session for which the callback was triggered.

is_export around here all we export is freedom.

keylength the length, in bits, of the ECCH key to be generated.

Returns:

a pointer to a ECDH key of proper size with parameters generated, or NULL on failure.

Definition at line 259 of file openssl.c.

References EC_GROUP_set_point_conversion_form_d, EC_KEY_get0_group_d, EC_KEY_new_by_curve_name_d, ecdh1024, ecdh512, and log_error.

Referenced by tls_server_create().

5.251.4.99 char* ssl_error_string (chr_t **buffer*, int_t *length*)

Get a textual representation of the last OpenSSL error message.

Parameters:

buffer a buffer that will receive the last OpenSSL error message.

length the size, in bytes, of the buffer that will contain the last OpenSSL error message.

Returns:

NULL on failure, or a pointer to the buffer where the last OpenSSL error message has been stored.

Definition at line 294 of file openssl.c.

References `ERR_error_string_n_d`, `ERR_get_error_d`, and `log_pedantic`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_group()`, `deprecated_ecies_key_alloc()`, `deprecated_ecies_key_create()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_private_bin()`, `deprecated_ecies_key_private_hex()`, `deprecated_ecies_key_public()`, `deprecated_ecies_key_public_bin()`, `deprecated_ecies_key_public_hex()`, `deprecated_symmetric_decrypt()`, `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_params_generate()`, `digest_id()`, `digest_length_output()`, `digest_name()`, `ecies_decrypt()`, `ecies_encrypt()`, `ecies_group()`, `ecies_key_alloc()`, `ecies_key_create()`, `ecies_key_private()`, `ecies_key_private_bin()`, `ecies_key_private_hex()`, `ecies_key_public()`, `ecies_key_public_bin()`, `ecies_key_public_hex()`, `ed25519_sign()`, `ed25519_verify()`, `hash_digest()`, `prime_start()`, `rand_get_int16()`, `rand_get_int32()`, `rand_get_int64()`, `rand_get_int8()`, `rand_get_uint16()`, `rand_get_uint32()`, `rand_get_uint64()`, `rand_get_uint8()`, `secp256k1_alloc()`, `secp256k1_compute_kek()`, `secp256k1_generate()`, `secp256k1_private_get()`, `secp256k1_private_set()`, `secp256k1_public_get()`, `secp256k1_public_set()`, `ssl_verify_privkey()`, `stacie_decrypt()`, `stacie_derive_key()`, `stacie_derive_seed()`, `stacie_derive_token()`, `stacie_realms_key()`, `symmetric_decrypt()`, `tls_client_alloc()`, and `tls_server_alloc()`.

5.251.4.100 void ssl_locking_callback(int mode, int n, const char *file, int line)

The locking function callback necessary for all multi-threaded applications using OpenSSL.

See also:

`CRYPTO_set_locking_callback()`

Parameters:

mode a bitmask specifying the requested openssl locking operation (`CRYPTO_LOCK`, `CRYPTO_WRITE`, `CRYPTO_READ`, or `CRYPTO_UNLOCK`).

n the zero-based index of the lock that is the target of the requested operation.

file a null-terminated string containing the filename of the function setting the lock, for debugging purposes.

line the line number of the function setting the lock, for debugging purposes.

Returns:

This function returns no value.

Definition at line 222 of file openssl.c.

References `mutex_lock()`, `mutex_unlock()`, and `ssl_locks`.

Referenced by `ssl_start()`.

5.251.4.101 bool_t ssl_start(void)

Initialize the OpenSSL facility.

Note:

First, this function initializes the mutexes necessary for the locking function callback that openssl uses for shared data structures in multi-threaded applications. Next, the DKIM key is retrieved if the [magma.dkim.enabled](#) configuration variable is set.

Returns:

true if openssl was initialized successfully, or false on failure.

Definition at line 114 of file openssl.c.

References `CRYPTO_num_locks_d`, `CRYPTO_set_id_callback_d`, `CRYPTO_set_locked_mem_functions_d`, `CRYPTO_set_locking_callback_d`, `dhparam_lock`, `log_critical`, `mm_alloc()`, `mm_sec_alloc()`, `mm_sec_free()`, `mutex_init()`, `OPENSSL_add_all_algorithms_noconf_d`, `SSL_library_init_d`, `SSL_load_error_strings_d`, `ssl_locking_callback()`, `ssl_locks`, and `ssl_thread_id_callback()`.

Referenced by `process_start()`.

5.251.4.102 void ssl_stop (void)

Stop the openssl library and cleanup all associated data structures.

Returns:

This function returns no values.

Definition at line 164 of file openssl.c.

References ASN1_STRING_TABLE_cleanup_d, BIO_sock_cleanup_d, COMP_zlib_cleanup_d, CONF_modules_unload_d, CRYPTO_cleanup_all_ex_data_d, CRYPTO_num_locks_d, CRYPTO_set_id_callback_d, CRYPTO_set_locking_callback_d, dhparam_lock, EC_KEY_free_d, ecdh1024, ecdh512, ENGINE_cleanup_d, ERR_free_strings_d, ERR_remove_thread_state_d, EVP_cleanup_d, mm_free(), mutex_destroy(), OBJ_cleanup_d, OBJ_NAME_cleanup_d, sk_pop_free_d, and ssl_locks.

Referenced by process_stop().

5.251.4.103 unsigned long ssl_thread_id_callback (void)

The thread id callback necessary for all multi-threaded applications using OpenSSL.

See also:

CRYPTO_set_id_callback()

Returns:

the id of the calling thread.

Definition at line 247 of file openssl.c.

References thread_get_thread_id().

Referenced by ssl_start().

5.251.4.104 void ssl_thread_stop (void)

Free the calling thread's SSL error queue.

See also:

[ssl_thread_stop\(\)](#)

Returns:

This function returns no value.

Definition at line 208 of file openssl.c.

References ERR_remove_thread_state_d.

Referenced by thread_stop().

5.251.4.105 bool_t ssl_verify_privkey (const char * *keyfile*)

Verify that the filename contains a valid private key in PEM format.

Parameters:

keyfile the pathname of the private key.

Returns:

true if the the key is valid, or false if it is not.

Definition at line 314 of file openssl.c.

References log_pedantic, MEMORYBUF, SSL_CTX_free_d, SSL_CTX_new_d, SSL_CTX_use_PrivateKey_file_d, ssl_error_string(), and SSLv23_client_method_d.

Referenced by dkim_start().

5.251.4.106 int_t tls_bits (TLS * *tls*)

[tls.c](#) [tls.c](#)

See also:

SSL_CIPHER_get_bits()

Definition at line 369 of file tls.c.

References SSL_CIPHER_get_bits_d, and SSL_get_current_cipher_d.

5.251.4.107 stringer_t* tls_cipher (TLS * *tls*, stringer_t * *output*)

Return the name of the TLS cipher suite assoicated with a connection.

See also:

SSL_CIPHER_get_name()

Parameters:

tls the TLS connection being inspected.

output a managed string to receive the encoded output; if passed as NULL, an output buffer will be allocated which must be freed by the caller.

Returns:

a pointer to the result, or NULL if an error occurs.

Definition at line 341 of file tls.c.

References NULLER, PLACER, SSL_CIPHER_get_name_d, SSL_get_current_cipher_d, SSL_get_version_d, st_cmp_cs_eq(), st_output(), and st_sprint().

Referenced by client_read(), client_read_line(), and client_write().

5.251.4.108 void* tls_client_alloc (int_t *sockd*)

Establish an TLS client wrapper around a socket descriptor.

Parameters:

sockd the file descriptor of the socket to have its transport security level upgraded.

Returns:

NULL on failure or a pointer to the SSL handle of the file descriptor if SSL negotiation was successful.

LOW: Add requisite config options and sandbox resources to verify server TLS certificates.

Definition at line 240 of file `tls.c`.

References `BIO_new_socket_d`, `ERR_clear_error_d`, `log_pedantic`, `MEMORYBUF`, `SSL_connect_d`, `SSL_CTX_ctrl_d`, `SSL_CTX_free_d`, `SSL_CTX_new_d`, `SSL_CTX_set_verify_d`, `ssl_error_string()`, `SSL_free_d`, `SSL_get_error_d`, `SSL_new_d`, `SSL_set_bio_d`, `SSL_set_connect_state_d`, and `SSLv23_client_method_d`.

Referenced by `client_secure()`.

5.251.4.109 `int tls_continue (TLS * tls, int result, int syserror)`

Return -1 if the connection is invalid, 0 if the operation should be retried, or a positive number indicating the number of bytes processed.

Definition at line 496 of file `tls.c`.

References `ERR_error_string_n_d`, `ERR_get_error_d`, `errno_name()`, `log_pedantic`, `MEMORYBUF`, `SSL_get_error_d`, and `status`.

Referenced by `tls_read()`, and `tls_write()`.

5.251.4.110 `stringer_t* tls_error (TLS * tls, int_t code, stringer_t * output)`

Consolidate the complicated logic associated with handling `SSL_read/SSL_write` calls which result in 0, or a negative number.

Definition at line 436 of file `tls.c`.

References `ERR_error_string_n_d`, `ERR_get_error_d`, `log_pedantic`, `MEMORYBUF`, `SSL_get_error_d`, `st_output()`, and `st_sprint()`.

Referenced by `client_read()`, `client_read_line()`, and `client_write()`.

5.251.4.111 `void tls_free (TLS * tls)`

Shutdown and free an TLS connection.

Parameters:

TLS the TLS connection to be shut down.

Returns:

This function returns no value.

Definition at line 316 of file `tls.c`.

References `ERR_remove_thread_state_d`, `log_pedantic`, `SSL_free_d`, and `SSL_shutdown_d`.

Referenced by `client_close()`, `con_destroy()`, and `protocol_secure()`.

5.251.4.112 `int tls_print (TLS * tls, const char * format, va_list args)`

Write formatted data to an TLS connection.

Parameters:

tls the SSL connection to which the data will be written.

format a format string specifying the data to be written to the SSL connection.

va_list a variable argument list containing the data parameters associated with the format string.

Returns:

-1 on error, or the number of bytes written to the SSL connection.

Definition at line 625 of file `tls.c`.

References `bytes`, `length`, `mm_alloc()`, `mm_free()`, `status`, and `tls_write()`.

5.251.4.113 `int tls_read (TLS * tls, void * buffer, int length, bool_t block)`

Read data from a TLS connection.

Parameters:

tls the TLS connection from which the data will be read.

buffer a pointer to the buffer where the read data will be stored.

length the length, in bytes, of the amount of data to be read.

block a boolean variable specifying whether the read operation should block.

Returns:

-1 on failure, 0 if the connection has been closed, or the number of bytes read from the connection on success.

Definition at line 556 of file `tls.c`.

References `ERR_clear_error_d`, `log_pedantic`, `SSL_read_d`, and `tls_continue()`.

Referenced by `client_read()`, `client_read_line()`, `con_read()`, and `con_read_line()`.

5.251.4.114 `TLS* tls_server_alloc (void * server, int sockd, int flags)`

Create a TLS session for a file descriptor, and accept the client TLS/SSL handshake.

See also:

`SSL_accept()`

`BIO_new_socket()`

Parameters:

server a server object which contains the underlying SSL context.

sockd the file descriptor of the TCP connection to be made SSL-ready.

flags passed to `BIO_new_socket()`, determines whether the socket is shut down when the BIO is freed.

Definition at line 158 of file `tls.c`.

References `BIO_new_socket_d`, `server_t::context`, `ERR_clear_error_d`, `errno_name()`, `log_pedantic`, `MEMORYBUF`, `SSL_accept_d`, `ssl_error_string()`, `SSL_free_d`, `SSL_get_error_d`, `SSL_new_d`, `SSL_set_accept_state_d`, `SSL_set_bio_d`, `status`, and `server_t::tls`.

Referenced by `imap_starttls()`, `pop_starttls()`, `protocol_secure()`, and `smtp_starttls()`.

5.251.4.115 `bool_t tls_server_create (void * server, uint_t security_level)`

Setup an TLS CTX for a server.

Note:

The server is passed as a void pointer because the provider layer doesn't comprehend protocol specific server instances.

Parameters:

server_t the TLS context will be assigned to the provided server context.

security_level an integer which will be used to control how the TLS context is configured: 0 = accept any type of SSL or TLS protocol version, and offer broad cipher support. 1 = require TLSv1 and above, refuse SSLv2 and SSLv3 connections, use any reasonably secure cipher. (recommened) 2 = require TLSv1 and above, refuse SSLv2 and SSLv3 connections, only use ciphers which provide forward secrecy. 3 = require TLSv1.2 and limit the cipher list to ECDHE-RSA-AES256-GCM-SHA384 or ECDHE-RSA-CHACHA20-POLY1305 as required by the specifications.

Definition at line 24 of file `tls.c`.

References `server_t::certificate`, `server_t::context`, `magma_t::cryptography`, `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_static_2048()`, `dh_static_4096()`, `magma_t::iface`, `log_critical`, `magma`, `MAGMA_CIPHERS_GENERIC`, `MAGMA_CIPHERS_HIGH`, `MAGMA_CIPHERS_MEDIUM`, `SSL_CTX_check_private_key_d`, `SSL_CTX_ctrl_d`, `SSL_CTX_new_d`, `SSL_CTX_set_cipher_list_d`, `SSL_CTX_set_tmp_dh_callback_d`, `SSL_CTX_set_tmp_ecdh_callback_d`, `SSL_CTX_set_verify_d`, `SSL_CTX_use_certificate_chain_file_d`, `SSL_CTX_use_PrivateKey_file_d`, `ssl_ecdh_exchange_callback()`, `SSLv23_server_method_d`, and `server_t::tls`.

Referenced by `servers_encryption_start()`.

5.251.4.116 `void tls_server_destroy (void * server)`

Destroy an TLS context associated with a server.

Parameters:

server the server to be deactivated.

Returns:

This function returns no value.

Definition at line 135 of file `tls.c`.

References `server_t::context`, `log_pedantic`, `SSL_CTX_free_d`, and `server_t::tls`.

Referenced by `servers_encryption_stop()`.

5.251.4.117 `int tls_status (TLS * tls)`

Checks whether a TLS connection has been shut down or not.

See also:

`SSL_get_shutdown()`

Parameters:

tls the TLS connection to be shut down.

Returns:

0 if the connection is alive and well, or `SSL_SENT_SHUTDOWN/SSL_RECEIVED_SHUTDOWN`

Definition at line 421 of file `tls.c`.

References `SSL_get_shutdown_d`.

Referenced by `client_status()`, and `con_status()`.

5.251.4.118 `chr_t* tls_suite (TLS * tls)`

Provide the SSL/TLS/DTLS cipher suite as a string constant.

See also:

SSL_CIPHER_get_name()

Note:

The RFC version of the TLS cipher suite is available through the SSL_CIPHER_standard_name() function.

Definition at line 403 of file tls.c.

References SSL_CIPHER_get_name_d, and SSL_get_current_cipher_d.

5.251.4.119 chr_t* tls_version (TLS *tls)

Provide the SSL/TLS/DTLS version as a string constant.

See also:

SSL_get_version()

Definition at line 385 of file tls.c.

References NULLER, PLACER, SSL_get_current_cipher_d, SSL_get_version_d, and st_cmp_cs_eq().

5.251.4.120 int tls_write (TLS *tls, const void *buffer, int length, bool_t block)

Write data to an open TLS connection.

Parameters:

tls the TLS connection to which the data will be written.

buffer a pointer to the buffer containing the data to be written.

length the length, in bytes, of the data to be written.

Returns:

-1 on error, or the number of bytes written to the TLS connection.

Definition at line 592 of file tls.c.

References ERR_clear_error_d, log_pedantic, SSL_write_d, and tls_continue().

Referenced by client_write(), con_write_bl(), and tls_print().

5.251.5 Variable Documentation**5.251.5.1 cryptex_head_t**

Definition at line 42 of file cryptography.h.

Referenced by cryptex_alloc(), cryptex_body_data(), cryptex_body_length(), cryptex_envelope_data(), cryptex_envelope_length(), cryptex_hmac_data(), cryptex_hmac_length(), cryptex_original_length(), cryptex_total_length(), deprecated_cryptex_alloc(), deprecated_cryptex_body_data(), deprecated_cryptex_body_length(), deprecated_cryptex_envelope_data(), deprecated_cryptex_envelope_length(), deprecated_cryptex_hmac_data(), deprecated_cryptex_hmac_length(), deprecated_cryptex_original_length(), and deprecated_cryptex_total_length().

5.251.5.2 `scramble_head_t`

Definition at line 59 of file `cryptography.h`.

Referenced by `deprecated_scramble_alloc()`, `deprecated_scramble_body_data()`, `deprecated_scramble_body_hash()`, `deprecated_scramble_body_length()`, `deprecated_scramble_decrypt()`, `deprecated_scramble_encrypt()`, `deprecated_scramble_import()`, `deprecated_scramble_orig_length()`, `deprecated_scramble_total_length()`, `deprecated_scramble_vector_data()`, `deprecated_scramble_vector_length()`, `scramble_alloc()`, `scramble_body_data()`, `scramble_body_hash()`, `scramble_body_length()`, `scramble_decrypt()`, `scramble_encrypt()`, `scramble_import()`, `scramble_orig_length()`, `scramble_total_length()`, `scramble_vector_data()`, and `scramble_vector_length()`.

5.252 src/providers/prime/cryptography/cryptography.h File Reference

Functions

- [placer_t aes_cipher_key](#) ([stringer_t](#) *key)

aes.c

- [placer_t aes_tag_shard](#) ([stringer_t](#) *key)

Extract the portion of the key used for the tag shard.

- [placer_t aes_vector_shard](#) ([stringer_t](#) *key)

Extract the portion of the key used for the vector shard.

- [stringer_t * aes_chunk_decrypt](#) ([stringer_t](#) *key, [stringer_t](#) *chunk, [stringer_t](#) *output)
- [stringer_t * aes_chunk_encrypt](#) ([uint8_t](#) type, [stringer_t](#) *key, [stringer_t](#) *chunk, [stringer_t](#) *output)

Create an encrypted chunk. The result includes the 1 byte type, the 3 byte big endian length, the 16 byte tag and the 16 byte vector, followed by the encrypted data. Note the caller must pad the data buffer to a 16 byte boundary.

- [stringer_t * aes_artifact_decrypt](#) ([stringer_t](#) *key, [stringer_t](#) *object, [stringer_t](#) *output)
- [stringer_t * aes_artifact_encrypt](#) ([stringer_t](#) *key, [stringer_t](#) *object, [stringer_t](#) *output)
- [ed25519_key_t * ed25519_alloc](#) (void)

ed25519.c

- void [ed25519_free](#) ([ed25519_key_t](#) *key)
- [ed25519_key_t * ed25519_generate](#) (void)
- [stringer_t * ed25519_private_get](#) ([ed25519_key_t](#) *key, [stringer_t](#) *output)
- [ed25519_key_t * ed25519_private_set](#) ([stringer_t](#) *key)
- [stringer_t * ed25519_public_get](#) ([ed25519_key_t](#) *key, [stringer_t](#) *output)
- [ed25519_key_t * ed25519_public_set](#) ([stringer_t](#) *key)
- [stringer_t * ed25519_sign](#) ([ed25519_key_t](#) *key, [stringer_t](#) *data, [stringer_t](#) *output)
- [ed25519_key_type_t ed25519_type](#) ([ed25519_key_t](#) *key)
- [int_t ed25519_verify](#) ([ed25519_key_t](#) *key, [stringer_t](#) *data, [stringer_t](#) *signature)

Verify an Ed25519 signature using the EdDSA algorithm.

- [secp256k1_key_t * secp256k1_alloc](#) (void)

secp256k1.c

- [stringer_t * secp256k1_compute_kek](#) ([secp256k1_key_t](#) *priv, [secp256k1_key_t](#) *pub, [stringer_t](#) *output)
- void [secp256k1_free](#) ([secp256k1_key_t](#) *key)

Free a secp256k1 key structure.

- [secp256k1_key_t * secp256k1_generate](#) (void)

Generate a random secp256k1 key pair.

- [stringer_t * secp256k1_private_get](#) ([secp256k1_key_t](#) *key, [stringer_t](#) *output)

Return a secp256k1 private key as a big endian integer inside a managed string.

- [secp256k1_key_t * secp256k1_private_set](#) ([stringer_t](#) *key)
- [stringer_t * secp256k1_public_get](#) ([secp256k1_key_t](#) *key, [stringer_t](#) *output)

Return a secp256k1 public key as a compressed point encoded as a big endian integer.

- [secp256k1_key_t * secp256k1_public_set](#) ([stringer_t](#) *key)
- [secp256k1_key_type_t secp256k1_type](#) ([secp256k1_key_t](#) *key)

5.252.1 Function Documentation

5.252.1.1 `stringer_t* aes_artifact_decrypt (stringer_t *key, stringer_t *object, stringer_t *output)`

Definition at line 627 of file aes.c.

References AES_BLOCK_LEN, aes_cipher_key(), AES_KEY_LEN, AES_TAG_LEN, aes_tag_shard(), AES_VECTOR_LEN, aes_vector_shard(), EVP_aes_256_gcm_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_key_length_d, EVP_DecryptFinal_ex_d, EVP_DecryptInit_ex_d, EVP_DecryptUpdate_d, log_pedantic, MANAGEDBUF, MEMORYBUF, mm_copy(), mm_move(), pl_data_get(), pl_empty(), pl_null(), PLACER, placer_t, prime_encrypted_object_header_t, prime_header_write(), PRIME_OBJECT_HEAD_LEN, PRIME_OBJECT_KEY_LEN, PRIME_OBJECT_PAYLOAD_PREFIX_LEN, PRIME_OBJECT_PREFIX_LEN, PRIME_ORG_KEY, PRIME_ORG_KEY_ENCRYPTED, PRIME_USER_KEY, PRIME_USER_KEY_ENCRYPTED, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), st_wipe(), st_xor(), and type().

Referenced by org_encrypted_key_set(), and user_encrypted_key_set().

5.252.1.2 `stringer_t* aes_artifact_encrypt (stringer_t *key, stringer_t *object, stringer_t *output)`

Definition at line 417 of file aes.c.

References AES_BLOCK_LEN, aes_cipher_key(), AES_KEY_LEN, AES_TAG_LEN, aes_tag_shard(), AES_VECTOR_LEN, aes_vector_shard(), EVP_aes_256_gcm_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_key_length_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, log_pedantic, MANAGEDBUF, MEMORYBUF, mm_copy(), pl_data_get(), pl_empty(), pl_null(), PLACER, placer_t, prime_encrypted_object_header_t, prime_header_read(), PRIME_OBJECT_HEAD_LEN, PRIME_OBJECT_KEY_LEN, PRIME_OBJECT_PAYLOAD_PREFIX_LEN, PRIME_ORG_KEY, PRIME_ORG_KEY_ENCRYPTED, PRIME_USER_KEY, PRIME_USER_KEY_ENCRYPTED, rand_write(), ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), st_wipe(), st_xor(), and type().

Referenced by org_encrypted_key_get(), and user_encrypted_key_get().

5.252.1.3 `stringer_t* aes_chunk_decrypt (stringer_t *key, stringer_t *chunk, stringer_t *output)`

Definition at line 275 of file aes.c.

References aes_cipher_key(), AES_KEY_LEN, AES_TAG_LEN, aes_tag_shard(), AES_VECTOR_LEN, aes_vector_shard(), EVP_aes_256_gcm_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_key_length_d, EVP_DecryptFinal_ex_d, EVP_DecryptInit_ex_d, EVP_DecryptUpdate_d, log_pedantic, MANAGEDBUF, MEMORYBUF, mm_copy(), pl_data_get(), pl_empty(), pl_null(), PLACER, placer_t, PRIME_CHUNK_HEAD_LEN, PRIME_CHUNK_KEY_LEN, PRIME_CHUNK_PREFIX_LEN, prime_encrypted_chunk_header_t, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), st_wipe(), and st_xor().

Referenced by encrypted_chunk_get().

5.252.1.4 `stringer_t* aes_chunk_encrypt (uint8_t type, stringer_t *key, stringer_t *chunk, stringer_t *output)`

Create an encrypted chunk. The result includes the 1 byte type, the 3 byte big endian length, the 16 byte tag and the 16 byte vector, followed by the encrypted data. Note the caller must pad the data buffer to a 16 byte boundary.

Definition at line 118 of file aes.c.

References AES_BLOCK_LEN, aes_cipher_key(), AES_KEY_LEN, AES_TAG_LEN, aes_tag_shard(), AES_VECTOR_LEN, aes_vector_shard(), EVP_aes_256_gcm_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_key_length_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, log_pedantic, MANAGEDBUF, MEMORYBUF, mm_copy(), pl_data_get(), pl_empty(), pl_null(), PLACER, placer_t, PRIME_CHUNK_HEAD_LEN, PRIME_CHUNK_KEY_LEN, prime_encrypted_chunk_header_t, rand_write(), ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), st_wipe(), and st_xor().

Referenced by encrypted_chunk_set().

5.252.1.5 `placer_t aes_cipher_key (stringer_t * key)`

[aes.c aes.c](#)

Parameters:

key The complete key, which holds the vector and tag shard values along with the cipher key.

Returns:

returns a place holder with the cipher key portion, or NULL if the supplied key was invalid.

Definition at line 58 of file `aes.c`.

References `AES_KEY_LEN`, `AES_TAG_LEN`, `AES_VECTOR_LEN`, `log_error`, `pl_init()`, `pl_null()`, `placer_t`, `PRIME_OBJECT_KEY_LEN`, `st_data_get()`, `st_empty`, and `st_length_get()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, and `aes_chunk_encrypt()`.

5.252.1.6 `placer_t aes_tag_shard (stringer_t * key)`

Extract the portion of the key used for the tag shard.

Parameters:

key The complete key, which holds the vector and tag shard values along with the cipher key.

Returns:

returns a placeholder with the tag shard, or NULL if the supplied key was invalid.

Definition at line 100 of file `aes.c`.

References `AES_TAG_LEN`, `AES_VECTOR_LEN`, `log_error`, `pl_init()`, `pl_null()`, `placer_t`, `PRIME_OBJECT_KEY_LEN`, `st_data_get()`, `st_empty`, and `st_length_get()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, and `aes_chunk_encrypt()`.

5.252.1.7 `placer_t aes_vector_shard (stringer_t * key)`

Extract the portion of the key used for the vector shard.

Parameters:

key The complete key, which holds the vector and tag shard values along with the cipher key.

Returns:

returns a place holder with the vector shard, or NULL if the supplied key was invalid.

Definition at line 79 of file `aes.c`.

References `AES_VECTOR_LEN`, `log_error`, `pl_init()`, `pl_null()`, `placer_t`, `PRIME_OBJECT_KEY_LEN`, `st_data_get()`, `st_empty`, and `st_length_get()`.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, and `aes_chunk_encrypt()`.

5.252.1.8 `ed25519_key_t* ed25519_alloc (void)`

[ed25519.c](#)

Definition at line 34 of file ed25519.c.

References ed25519_key_t, log_pedantic, mm_alloc(), and mm_wipe().

Referenced by ed25519_generate(), ed25519_private_set(), and ed25519_public_set().

5.252.1.9 void ed25519_free (ed25519_key_t * key)

Definition at line 19 of file ed25519.c.

References log_pedantic, and mm_free().

Referenced by encrypted_message_free(), ephemeral_chunk_free(), org_key_free(), org_signet_free(), user_key_free(), and user_signet_free().

5.252.1.10 ed25519_key_t* ed25519_generate (void)

Definition at line 48 of file ed25519.c.

References ed25519_alloc(), ED25519_KEY_PRIV_LEN, ED25519_KEY_PUB_LEN, ed25519_key_t, ED25519_keypair_d, ED25519_PRIV, and mm_copy().

Referenced by naked_message_set(), org_key_generate(), and user_key_generate().

5.252.1.11 stringer_t* ed25519_private_get (ed25519_key_t * key, stringer_t * output)

Definition at line 90 of file ed25519.c.

References ED25519_KEY_PRIV_LEN, ED25519_PRIV, log_pedantic, mm_copy(), st_alloc(), st_avail_get(), st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and st_wipe().

Referenced by org_key_get(), and user_key_get().

5.252.1.12 ed25519_key_t* ed25519_private_set (stringer_t * key)

Definition at line 132 of file ed25519.c.

References ed25519_alloc(), ED25519_KEY_PRIV_LEN, ed25519_key_t, ED25519_keypair_from_seed_d, ED25519_PRIV, log_info, mm_copy(), st_data_get(), st_empty, and st_length_get().

Referenced by org_key_set(), and user_key_set().

5.252.1.13 stringer_t* ed25519_public_get (ed25519_key_t * key, stringer_t * output)

Definition at line 65 of file ed25519.c.

References ED25519_KEY_PUB_LEN, ED25519_PRIV, ED25519_PUB, log_pedantic, mm_copy(), st_alloc(), st_avail_get(), st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and st_wipe().

Referenced by ephemeral_chunk_get(), org_signet_fingerprint(), org_signet_generate(), org_signet_get(), org_signet_verify(), user_request_generate(), user_request_get(), user_request_rotation(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_get(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.252.1.14 ed25519_key_t* ed25519_public_set (stringer_t * key)

Definition at line 114 of file ed25519.c.

References ed25519_alloc(), ED25519_KEY_PUB_LEN, ed25519_key_t, ED25519_PUB, log_info, mm_copy(), st_data_get(), st_empty, and st_length_get().

Referenced by `ephemeral_chunk_get()`, `ephemeral_chunk_set()`, `org_signet_generate()`, `org_signet_set()`, `user_request_generate()`, `user_request_rotation()`, `user_request_set()`, `user_request_sign()`, and `user_signet_set()`.

5.252.1.15 `stringer_t* ed25519_sign (ed25519_key_t *key, stringer_t *data, stringer_t *output)`

Definition at line 151 of file `ed25519.c`.

References `ED25519_PRIV`, `ED25519_sign_d`, `ED25519_SIGNATURE_LEN`, `log_info`, `log_pedantic`, `MEMORYBUF`, `ssl_error_string()`, `st_alloc()`, `st_avail_get()`, `st_cleanup`, `st_data_get()`, `st_empty`, `st_length_get()`, `st_length_set()`, `st_opt_get()`, `st_valid_destination()`, and `st_valid_tracked()`.

5.252.1.16 `ed25519_key_type_t ed25519_type (ed25519_key_t *key)`

Definition at line 11 of file `ed25519.c`.

References `ED25519_ERR`.

Referenced by `encrypted_chunk_get()`, `encrypted_chunk_set()`, `part_decrypt()`, `part_encrypt()`, `signature_full_get()`, `signature_full_verify()`, `signature_tree_get()`, and `signature_tree_verify()`.

5.252.1.17 `int_t ed25519_verify (ed25519_key_t *key, stringer_t *data, stringer_t *signature)`

Verify an Ed25519 signature using the EdDSA algorithm.

Parameters:

key The public signing key.

data The data being verified.

signature The signature.

Returns:

0 for successful signature verification, -1 for a signature verification failures, -2 for processing or parameter issue.

Definition at line 196 of file `ed25519.c`.

References `ED25519_PRIV`, `ED25519_PUB`, `ED25519_SIGNATURE_LEN`, `ed25519_verify()`, `ED25519_verify_d`, `log_info`, `log_pedantic`, `MEMORYBUF`, `ssl_error_string()`, `st_data_get()`, `st_empty`, and `st_length_get()`.

Referenced by `ed25519_sign_open()`, `ed25519_verify()`, `encrypted_chunk_get()`, `org_signet_verify()`, `signature_full_verify()`, `signature_tree_verify()`, `user_request_verify_chain_of_custody()`, `user_request_verify_self()`, `user_signet_verify_chain_of_custody()`, `user_signet_verify_org()`, and `user_signet_verify_self()`.

5.252.1.18 `secp256k1_key_t* secp256k1_alloc (void)`

[secp256k1.c](#) [secp256k1.c](#)

See also:

`NID_secp256k1`

Returns:

NULL on failure, or a pointer to the newly allocated key pair.

Definition at line 53 of file `secp256k1.c`.

References `EC_KEY_free_d`, `EC_KEY_new_by_curve_name_d`, `EC_KEY_new_d`, `EC_KEY_set_conv_form_d`, `EC_KEY_set_group_d`, `log_info`, `MEMORYBUF`, `prime_curve_group`, and `ssl_error_string()`.

Referenced by `secp256k1_generate()`, `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.252.1.19 stringer_t* secp256k1_compute_kek (secp256k1_key_t * *priv*, secp256k1_key_t * *pub*, stringer_t * *output*)

Parameters:

private

public

output

Returns:

Definition at line 321 of file secp256k1.c.

References EC_KEY_get0_public_key_d, ECDH_compute_key_d, log_info, log_pedantic, MEMORYBUF, SECP256K1_SHARED_SECRET_LEN, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), and st_valid_tracked().

Referenced by keks_get(), and keks_set().

5.252.1.20 void secp256k1_free (secp256k1_key_t * *key*)

Free a secp256k1 key structure.

See also:

EC_KEY_free()

Parameters:

key the managed string to be freed.

Returns:

This function returns no value.

Definition at line 33 of file secp256k1.c.

References EC_KEY_free_d, and log_pedantic.

Referenced by encrypted_message_free(), ephemeral_chunk_free(), org_key_free(), org_signet_free(), user_key_free(), and user_signet_free().

5.252.1.21 secp256k1_key_t* secp256k1_generate (void)

Generate a random secp256k1 key pair.

Returns:

NULL on failure, or a new, randomly generated secp256k1 key pair on success.

Definition at line 86 of file secp256k1.c.

References EC_KEY_free_d, EC_KEY_generate_key_d, log_info, MEMORYBUF, secp256k1_alloc(), and ssl_error_string().

Referenced by _generate_ec_keypair(), naked_message_set(), org_key_generate(), and user_key_generate().

5.252.1.22 stringer_t* secp256k1_private_get (secp256k1_key_t * *key*, stringer_t * *output*)

Return a secp256k1 private key as a big endian integer inside a managed string.

Parameters:

key the input secp256k1 key pair.

Returns:

NULL on failure, or the private key as a big endian integer.

Definition at line 151 of file secp256k1.c.

References BN_bn2bin_d, BN_num_bits_d, BN_num_bytes_d, EC_KEY_get0_private_key_d, log_pedantic, MEMORYBUF, SECP256K1_KEY_PRIV_LEN, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and st_wipe().

Referenced by _serialize_ec_privkey(), org_key_get(), and user_key_get().

5.252.1.23 secp256k1_key_t* secp256k1_private_set (stringer_t * *key*)**Parameters:**

key

Returns:

Definition at line 257 of file secp256k1.c.

References BN_bin2bn_d, BN_CTX_free_d, BN_CTX_new_d, BN_CTX_start_d, BN_free_d, EC_KEY_check_key_d, EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_set_private_key_d, EC_KEY_set_public_key_d, EC_POINT_free_d, EC_POINT_mul_d, EC_POINT_new_d, log_info, MEMORYBUF, number, secp256k1_alloc(), SECP256K1_KEY_PRIV_LEN, ssl_error_string(), st_data_get(), st_empty, and st_length_get().

Referenced by _deserialize_ec_privkey(), _load_ec_privkey(), org_key_set(), and user_key_set().

5.252.1.24 stringer_t* secp256k1_public_get (secp256k1_key_t * *key*, stringer_t * *output*)

Return a secp256k1 public key as a compressed point encoded as a big endian integer.

Parameters:

key the input secp256k1 key pair.

Returns:

NULL on failure, or the public key as a big endian integer.

Definition at line 110 of file secp256k1.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_KEY_get_conv_form_d, EC_POINT_point2oct_d, log_pedantic, MEMORYBUF, SECP256K1_KEY_PUB_LEN, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), and st_valid_tracked().

Referenced by _serialize_ec_pubkey(), ephemeral_chunk_get(), naked_message_set(), org_signet_fingerprint(), org_signet_generate(), org_signet_get(), org_signet_verify(), user_request_generate(), user_request_get(), user_request_rotation(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_get(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.252.1.25 secp256k1_key_t* secp256k1_public_set (stringer_t * *key*)**Parameters:**

key

Returns:

Definition at line 205 of file secp256k1.c.

References BN_CTX_free_d, BN_CTX_new_d, BN_CTX_start_d, EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_set_public_key_d, EC_POINT_free_d, EC_POINT_new_d, EC_POINT_oct2point_d, log_info, MEMORYBUF, secp256k1_alloc(), SECP256K1_KEY_PUB_LEN, ssl_error_string(), st_char_get(), st_data_get(), st_empty, and st_length_get().

Referenced by ephemeral_chunk_get(), ephemeral_chunk_set(), naked_message_set(), org_signet_generate(), org_signet_set(), user_request_generate(), user_request_rotation(), user_request_set(), user_request_sign(), and user_signet_set().

5.252.1.26 secp256k1_key_type_t secp256k1_type (secp256k1_key_t * *key*)

Definition at line 12 of file secp256k1.c.

References EC_KEY_get0_private_key_d, EC_KEY_get0_public_key_d, SECP256K1_ERR, SECP256K1_PRIV, and SECP256K1_PUB.

Referenced by keks_get(), and keks_set().

5.253 src/providers/cryptography/digest.c File Reference

```
#include "magma.h"
```

Functions

- [digest_t * digest_name](#) ([stringer_t](#) *name)
- [digest_t * digest_id](#) ([int_t](#) id)
digest.c
- [int_t digest_length_output](#) (const [digest_t](#) *digest)

5.253.1 Function Documentation

5.253.1.1 [digest_t * digest_id](#) ([int_t](#) id)

[digest.c](#)

Definition at line 21 of file [digest.c](#).

References [EVP_get_digestbyname_d](#), [log_pedantic](#), [MEMORYBUF](#), [OBJ_nid2sn_d](#), and [ssl_error_string\(\)](#).

5.253.1.2 [int_t digest_length_output](#) (const [digest_t](#) * *digest*)

Definition at line 32 of file [digest.c](#).

References [EVP_MD_size_d](#), [EVP_MD_type_d](#), [log_pedantic](#), [MEMORYBUF](#), [OBJ_nid2sn_d](#), and [ssl_error_string\(\)](#).

5.253.1.3 [digest_t * digest_name](#) ([stringer_t](#) * *name*)

Definition at line 10 of file [digest.c](#).

References [EVP_get_digestbyname_d](#), [log_pedantic](#), [MEMORYBUF](#), [ssl_error_string\(\)](#), [st_char_get\(\)](#), [st_empty](#), and [st_length_int\(\)](#).

5.254 src/providers/cryptography/ecies.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t deprecated_ecies_start](#) (void)
- void [deprecated_ecies_stop](#) (void)
Destroy all initialized ECIES curve group information.
- void [deprecated_ecies_key_free](#) (EC_KEY *key)
Free an ECIES key pair.
- EC_KEY * [deprecated_ecies_key_alloc](#) (void)
Allocate a new ECIES key pair from the curve defined by ECIES_CURVE.
- EC_GROUP * [deprecated_ecies_group](#) (uint64_t curve, [bool_t](#) precompute)
- EC_KEY * [deprecated_ecies_key_create](#) (void)
Generate a random ECIES key pair.
- EC_KEY * [deprecated_ecies_key_public](#) (uint64_t format, [placer_t](#) data)
- EC_KEY * [deprecated_ecies_key_private](#) (uint64_t format, [placer_t](#) data)
- [stringer_t](#) * [deprecated_ecies_key_public_hex](#) (EC_KEY *key)
Return an ECIES public key as a null-terminated hex string.
- [uchr_t](#) * [deprecated_ecies_key_public_bin](#) (EC_KEY *key, [size_t](#) *olen)
Return an ECIES public key as binary data.
- [stringer_t](#) * [deprecated_ecies_key_private_hex](#) (EC_KEY *key)
Return an ECIES private key as a hex string.
- [uchr_t](#) * [deprecated_ecies_key_private_bin](#) (EC_KEY *key, [size_t](#) *olen)
Return an ECIES private key as binary data.
- void * [deprecated_ecies_envelope_derivation](#) (const void *input, [size_t](#) ilen, void *output, [size_t](#) *olen)
- [cryptex_t](#) * [deprecated_ecies_encrypt](#) ([stringer_t](#) *key, [ECIES_KEY_TYPE](#) key_type, unsigned char *data, [size_t](#) length)
Encrypt a block of data using an ECIES public key.
- [uchr_t](#) * [deprecated_ecies_decrypt](#) ([stringer_t](#) *key, [ECIES_KEY_TYPE](#) key_type, [cryptex_t](#) *cryptex, [size_t](#) *length)
Decrypt a block of data using an ECIES private key.

Variables

- EC_GROUP * [ecies_curve_group](#) = NULL
- const EVP_MD * [ecies_hmac_evp](#) = NULL
- const EVP_MD * [ecies_envelope_evp](#) = NULL
- const EVP_CIPHER * [ecies_cipher_evp](#) = NULL

5.254.1 Function Documentation

5.254.1.1 `uchr_t* deprecated_ecies_decrypt (stringer_t *key, ECIES_KEY_TYPE key_type, cryptex_t *cryptex, size_t *length)`

Decrypt a block of data using an ECIES private key. `ecies.c`

Parameters:

- key** the ECIES private key in the specified format.
- key_type** the encoding type of the ECIES private key (ECIES_PRIVATE_BINARY or ECIES_PRIVATE_HEX).
- cryptex** a pointer to the head of the cryptex object with the encrypted data.
- length** a pointer to a `size_t` variable which will receive the final length of the unencrypted data.

Returns:

NULL on failure, or a pointer to a memory address containing the decrypted data on success..

Definition at line 578 of file `ecies.c`.

References `deprecated_cryptex_body_data()`, `deprecated_cryptex_body_length()`, `deprecated_cryptex_envelope_data()`, `deprecated_cryptex_envelope_length()`, `deprecated_cryptex_hmac_data()`, `deprecated_cryptex_hmac_length()`, `deprecated_cryptex_original_length()`, `deprecated_ecies_envelope_derivation()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_public()`, `EC_KEY_free_d`, `EC_KEY_get0_public_key_d`, `ECDH_compute_key_d`, `ECIES_CIPHER`, `ECIES_HMAC`, `ECIES_PRIVATE_BINARY`, `ECIES_PRIVATE_HEX`, `ECIES_PUBLIC_BINARY`, `EVP_CIPHER_CTX_cleanup_d`, `EVP_CIPHER_CTX_init_d`, `EVP_CIPHER_CTX_set_padding_d`, `EVP_CIPHER_key_length_d`, `EVP_DecryptFinal_ex_d`, `EVP_DecryptInit_ex_d`, `EVP_DecryptUpdate_d`, `EVP_get_cipherbyname_d`, `EVP_get_digestbyname_d`, `HMAC_CTX_cleanup_d`, `HMAC_CTX_init_d`, `HMAC_Final_d`, `HMAC_Init_ex_d`, `HMAC_Update_d`, `log_info`, `MEMORYBUF`, `mm_alloc()`, `OBJ_nid2sn_d`, `pl_init()`, `ssl_error_string()`, and `st_empty_out()`.

5.254.1.2 `cryptex_t* deprecated_ecies_encrypt (stringer_t *key, ECIES_KEY_TYPE key_type, unsigned char *data, size_t length)`

Encrypt a block of data using an ECIES public key.

Parameters:

- key** the ECIES public key in the specified format.
- key_type** the encoding type of the ECIES public key (ECIES_PUBLIC_BINARY or ECIES_PUBLIC_HEX).
- data** a pointer to the block of data to be encrypted.
- length** the length, in bytes, of the data to be encrypted.

Returns:

NULL on failure, or a pointer to the header of the cryptex object containing the encrypted data on success..

Definition at line 407 of file `ecies.c`.

References `deprecated_cryptex_alloc()`, `deprecated_cryptex_body_data()`, `deprecated_cryptex_body_length()`, `deprecated_cryptex_envelope_data()`, `deprecated_cryptex_free()`, `deprecated_cryptex_hmac_data()`, `deprecated_cryptex_hmac_length()`, `deprecated_ecies_envelope_derivation()`, `deprecated_ecies_key_create()`, `deprecated_ecies_key_public()`, `EC_KEY_free_d`, `EC_KEY_get0_group_d`, `EC_KEY_get0_public_key_d`, `EC_POINT_point2oct_d`, `ECDH_compute_key_d`, `ECIES_CIPHER`, `ECIES_HMAC`, `ECIES_PUBLIC_BINARY`, `ECIES_PUBLIC_HEX`, `EVP_CIPHER_block_size_d`, `EVP_CIPHER_CTX_cleanup_d`, `EVP_CIPHER_CTX_init_d`, `EVP_CIPHER_CTX_set_padding_d`, `EVP_CIPHER_key_length_d`, `EVP_EncryptFinal_ex_d`, `EVP_EncryptInit_ex_d`, `EVP_EncryptUpdate_d`, `EVP_get_cipherbyname_d`, `EVP_get_digestbyname_d`, `EVP_MD_size_d`, `HMAC_CTX_cleanup_d`, `HMAC_CTX_init_d`, `HMAC_Final_d`, `HMAC_Init_ex_d`, `HMAC_Update_d`, `log_info`, `MEMORYBUF`, `OBJ_nid2sn_d`, `pl_init()`, `ssl_error_string()`, and `st_empty_out()`.

5.254.1.3 void* deprecated_ecies_envelope_derivation (const void * *input*, size_t *ilen*, void * *output*, size_t * *olen*)

Definition at line 390 of file ecies.c.

References ecies_envelope_evpc, and EVP_Digest_d.

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.254.1.4 EC_GROUP* deprecated_ecies_group (uint64_t *curve*, bool_t *precompute*)

Definition at line 89 of file ecies.c.

References EC_GROUP_free_d, EC_GROUP_new_by_curve_name_d, EC_GROUP_precompute_mult_d, EC_GROUP_set_point_conversion_form_d, log_error, MEMORYBUF, and ssl_error_string().

Referenced by deprecated_ecies_start().

5.254.1.5 EC_KEY* deprecated_ecies_key_alloc (void)

Allocate a new ECIES key pair from the curve defined by ECIES_CURVE.

See also:

NID_sect571k1

Returns:

NULL on failure, or a pointer to the newly allocated key pair.

Definition at line 58 of file ecies.c.

References EC_KEY_free_d, EC_KEY_new_by_curve_name_d, EC_KEY_new_d, EC_KEY_set_conv_form_d, EC_KEY_set_group_d, ECIES_CURVE, ecies_curve_group, log_info, MEMORYBUF, and ssl_error_string().

Referenced by deprecated_ecies_key_create(), deprecated_ecies_key_private(), and deprecated_ecies_key_public().

5.254.1.6 EC_KEY* deprecated_ecies_key_create (void)

Generate a random ECIES key pair.

Returns:

NULL on failure, or a new random ECIES key pair on success.

Definition at line 112 of file ecies.c.

References deprecated_ecies_key_alloc(), EC_KEY_free_d, EC_KEY_generate_key_d, log_info, MEMORYBUF, and ssl_error_string().

Referenced by deprecated_ecies_encrypt().

5.254.1.7 void deprecated_ecies_key_free (EC_KEY * *key*)

Free an ECIES key pair.

Returns:

This function returns no value.

Definition at line 47 of file ecies.c.

References EC_KEY_free_d.

5.254.1.8 EC_KEY* deprecated_ecies_key_private (uint64_t format, placer_t data)

Definition at line 199 of file ecies.c.

References BN_bin2bn_d, BN_free_d, BN_hex2bn_d, deprecated_ecies_key_alloc(), EC_KEY_free_d, EC_KEY_set_private_key_d, ECIES_PRIVATE_BINARY, ECIES_PRIVATE_HEX, log_info, MEMORYBUF, number, pl_char_get(), pl_data_get(), pl_length_get(), and ssl_error_string().

Referenced by deprecated_ecies_decrypt().

5.254.1.9 uchr_t* deprecated_ecies_key_private_bin (EC_KEY * key, size_t * olen)

Return an ECIES private key as binary data.

Parameters:

key the input ECIES key pair.

olen a pointer to store the length of the returned key.

Returns:

NULL on failure, or a pointer to the raw private key.

Definition at line 359 of file ecies.c.

References BN_bn2bin_d, BN_num_bytes_d, EC_KEY_get0_private_key_d, log_info, MEMORYBUF, mm_sec_alloc(), mm_sec_free(), and ssl_error_string().

5.254.1.10 stringer_t* deprecated_ecies_key_private_hex (EC_KEY * key)

Return an ECIES private key as a hex string.

Parameters:

key the input ECIES key pair.

Returns:

NULL on failure, or the hex-formatted private key as a managed string.

Definition at line 325 of file ecies.c.

References BN_bn2hex_d, CONTIGUOUS, EC_KEY_get0_private_key_d, log_pedantic, MANAGED_T, MEMORYBUF, ns_length_get(), ns_wipe(), OPENSSL_free_d, SECURE, ssl_error_string(), and st_import_opts().

5.254.1.11 EC_KEY* deprecated_ecies_key_public (uint64_t format, placer_t data)

Definition at line 131 of file ecies.c.

References deprecated_ecies_key_alloc(), EC_KEY_check_key_d, EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_set_public_key_d, EC_POINT_free_d, EC_POINT_hex2point_d, EC_POINT_new_d, EC_POINT_oct2point_d, ECIES_PUBLIC_BINARY, ECIES_PUBLIC_HEX, log_info, MEMORYBUF, pl_char_get(), pl_data_get(), pl_length_get(), and ssl_error_string().

Referenced by deprecated_ecies_decrypt(), and deprecated_ecies_encrypt().

5.254.1.12 uchr_t* deprecated_ecies_key_public_bin (EC_KEY * key, size_t * olen)

Return an ECIES public key as binary data.

Parameters:

key the input ECIES key pair.

olen a pointer to store the length of the returned key.

Returns:

NULL on failure, or a pointer to the raw public key.

Definition at line 288 of file ecies.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2oct_d, log_info, MEMORYBUF, mm_alloc(), mm_free(), and ssl_error_string().

5.254.1.13 stringer_t* deprecated_ecies_key_public_hex (EC_KEY * key)

Return an ECIES public key as a null-terminated hex string.

Parameters:

key the input ECIES key pair.

Returns:

NULL on failure, or the hex-formatted public key as a null-terminated string.

Definition at line 255 of file ecies.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2hex_d, log_info, MEMORYBUF, ns_length_get(), OPENSSL_free_d, ssl_error_string(), and st_import().

5.254.1.14 bool_t deprecated_ecies_start (void)

Definition at line 15 of file ecies.c.

References deprecated_ecies_group(), ECIES_CIPHER, ecies_cipher_evp, ECIES_CURVE, ecies_curve_group, ECIES_ENVELOPE, ecies_envelope_evp, ECIES_HMAC, ecies_hmac_evp, EVP_get_cipherbyname_d, EVP_get_digestbyname_d, log_error, and OBJ_nid2sn_d.

Referenced by process_start().

5.254.1.15 void deprecated_ecies_stop (void)

Destroy all initialized ECIES curve group information.

Returns:

This function returns no value.

Definition at line 30 of file ecies.c.

References EC_GROUP_free_d, and ecies_curve_group.

Referenced by process_stop().

5.254.2 Variable Documentation**5.254.2.1 const EVP_CIPHER* ecies_cipher_evp = NULL**

Definition at line 13 of file ecies.c.

Referenced by deprecated_ecies_start(), and ecies_start().

5.254.2.2 EC_GROUP* ecies_curve_group = NULL

Definition at line 10 of file ecies.c.

Referenced by deprecated_ecies_key_alloc(), deprecated_ecies_start(), deprecated_ecies_stop(), ecies_key_alloc(), ecies_start(), and ecies_stop().

5.254.2.3 const EVP_MD* ecies_envelope_evp = NULL

Definition at line 12 of file ecies.c.

Referenced by deprecated_ecies_envelope_derivation(), deprecated_ecies_start(), ecies_envelope_derivation(), and ecies_start().

5.254.2.4 const EVP_MD* ecies_hmac_evp = NULL

Definition at line 11 of file ecies.c.

Referenced by deprecated_ecies_start(), and ecies_start().

5.255 src/providers/deprecated/ecies.c File Reference

```
#include "magma.h"
#include "deprecated.h"
```

Functions

- `bool_t ecies_start` (void)
- `void ecies_stop` (void)
Destroy all initialized ECIES curve group information.
- `void ecies_key_free` (EC_KEY *key)
Free an ECIES key pair.
- `EC_KEY * ecies_key_alloc` (void)
Allocate a new ECIES key pair from the curve defined by ECIES_CURVE.
- `EC_GROUP * ecies_group` (uint64_t curve, `bool_t` precompute)
- `EC_KEY * ecies_key_create` (void)
Generate a random ECIES key pair.
- `EC_KEY * ecies_key_public` (uint64_t format, `placer_t` data)
- `EC_KEY * ecies_key_private` (uint64_t format, `placer_t` data)
- `stringer_t * ecies_key_public_hex` (EC_KEY *key)
Return an ECIES public key as a null-terminated hex string.
- `uchr_t * ecies_key_public_bin` (EC_KEY *key, `size_t` *olen)
Return an ECIES public key as binary data.
- `stringer_t * ecies_key_private_hex` (EC_KEY *key)
Return an ECIES private key as a hex string.
- `uchr_t * ecies_key_private_bin` (EC_KEY *key, `size_t` *olen)
Return an ECIES private key as binary data.
- `void * ecies_envelope_derivation` (const void *input, `size_t` ilen, void *output, `size_t` *olen)
- `cryptex_t * ecies_encrypt` (`stringer_t` *key, ECIES_KEY_TYPE key_type, unsigned char *data, `size_t` length)
Encrypt a block of data using an ECIES public key.
- `uchr_t * ecies_decrypt` (`stringer_t` *key, ECIES_KEY_TYPE key_type, `cryptex_t` *cryptex, `size_t` *length)
Decrypt a block of data using an ECIES private key.

Variables

- EC_GROUP * `ecies_curve_group`
- const EVP_MD * `ecies_hmac_evp`
- const EVP_MD * `ecies_envelope_evp`
- const EVP_CIPHER * `ecies_cipher_evp`

5.255.1 Function Documentation

5.255.1.1 `uchr_t* ecies_decrypt (stringer_t *key, ECIES_KEY_TYPE key_type, cryptex_t *cryptex, size_t *length)`

Decrypt a block of data using an ECIES private key. `ecies.c`

Parameters:

- key* the ECIES private key in the specified format.
- key_type* the encoding type of the ECIES private key (ECIES_PRIVATE_BINARY or ECIES_PRIVATE_HEX).
- cryptex* a pointer to the head of the cryptex object with the encrypted data.
- length* a pointer to a `size_t` variable which will receive the final length of the unencrypted data.

Returns:

NULL on failure, or a pointer to a memory address containing the decrypted data on success..

Definition at line 579 of file `ecies.c`.

References `cryptex_body_data()`, `cryptex_body_length()`, `cryptex_envelope_data()`, `cryptex_envelope_length()`, `cryptex_hmac_data()`, `cryptex_hmac_length()`, `cryptex_original_length()`, `EC_KEY_free_d`, `EC_KEY_get0_public_key_d`, `ECDH_compute_key_d`, `ECIES_CIPHER`, `ecies_envelope_derivation()`, `ECIES_HMAC`, `ecies_key_private()`, `ecies_key_public()`, `ECIES_PRIVATE_BINARY`, `ECIES_PRIVATE_HEX`, `ECIES_PUBLIC_BINARY`, `EVP_CIPHER_CTX_cleanup_d`, `EVP_CIPHER_CTX_init_d`, `EVP_CIPHER_CTX_set_padding_d`, `EVP_CIPHER_key_length_d`, `EVP_DecryptFinal_ex_d`, `EVP_DecryptInit_ex_d`, `EVP_DecryptUpdate_d`, `EVP_get_cipherbyname_d`, `EVP_get_digestbyname_d`, `HMAC_CTX_cleanup_d`, `HMAC_CTX_init_d`, `HMAC_Final_d`, `HMAC_Init_ex_d`, `HMAC_Update_d`, `log_info`, `MEMORYBUF`, `mm_alloc()`, `OBJ_nid2sn_d`, `pl_init()`, `ssl_error_string()`, and `st_empty_out()`.

5.255.1.2 `cryptex_t* ecies_encrypt (stringer_t *key, ECIES_KEY_TYPE key_type, unsigned char *data, size_t length)`

Encrypt a block of data using an ECIES public key.

Parameters:

- key* the ECIES public key in the specified format.
- key_type* the encoding type of the ECIES public key (ECIES_PUBLIC_BINARY or ECIES_PUBLIC_HEX).
- data* a pointer to the block of data to be encrypted.
- length* the length, in bytes, of the data to be encrypted.

Returns:

NULL on failure, or a pointer to the header of the cryptex object containing the encrypted data on success..

Definition at line 408 of file `ecies.c`.

References `cryptex_alloc()`, `cryptex_body_data()`, `cryptex_body_length()`, `cryptex_envelope_data()`, `cryptex_free()`, `cryptex_hmac_data()`, `cryptex_hmac_length()`, `EC_KEY_free_d`, `EC_KEY_get0_group_d`, `EC_KEY_get0_public_key_d`, `EC_POINT_point2oct_d`, `ECDH_compute_key_d`, `ECIES_CIPHER`, `ecies_envelope_derivation()`, `ECIES_HMAC`, `ecies_key_create()`, `ecies_key_public()`, `ECIES_PUBLIC_BINARY`, `ECIES_PUBLIC_HEX`, `EVP_CIPHER_block_size_d`, `EVP_CIPHER_CTX_cleanup_d`, `EVP_CIPHER_CTX_init_d`, `EVP_CIPHER_CTX_set_padding_d`, `EVP_CIPHER_key_length_d`, `EVP_EncryptFinal_ex_d`, `EVP_EncryptInit_ex_d`, `EVP_EncryptUpdate_d`, `EVP_get_cipherbyname_d`, `EVP_get_digestbyname_d`, `EVP_MD_size_d`, `HMAC_CTX_cleanup_d`, `HMAC_CTX_init_d`, `HMAC_Final_d`, `HMAC_Init_ex_d`, `HMAC_Update_d`, `log_info`, `MEMORYBUF`, `OBJ_nid2sn_d`, `pl_init()`, `ssl_error_string()`, and `st_empty_out()`.

5.255.1.3 `void* ecies_envelope_derivation (const void *input, size_t ilen, void *output, size_t *olen)`

Definition at line 391 of file `ecies.c`.

References `ecies_envelope_evp`, and `EVP_Digest_d`.

Referenced by `ecies_decrypt()`, and `ecies_encrypt()`.

5.255.1.4 EC_GROUP* ecies_group (uint64_t curve, bool_t precompute)

Definition at line 90 of file ecies.c.

References EC_GROUP_free_d, EC_GROUP_new_by_curve_name_d, EC_GROUP_precompute_mult_d, EC_GROUP_set_point_conversion_form_d, log_error, MEMORYBUF, and ssl_error_string().

Referenced by ecies_start().

5.255.1.5 EC_KEY* ecies_key_alloc (void)

Allocate a new ECIES key pair from the curve defined by ECIES_CURVE.

See also:

NID_sect571k1

Returns:

NULL on failure, or a pointer to the newly allocated key pair.

Definition at line 59 of file ecies.c.

References EC_KEY_free_d, EC_KEY_new_by_curve_name_d, EC_KEY_new_d, EC_KEY_set_conv_form_d, EC_KEY_set_group_d, ECIES_CURVE, ecies_curve_group, log_info, MEMORYBUF, and ssl_error_string().

Referenced by ecies_key_create(), ecies_key_private(), and ecies_key_public().

5.255.1.6 EC_KEY* ecies_key_create (void)

Generate a random ECIES key pair.

Returns:

NULL on failure, or a new random ECIES key pair on success.

Definition at line 113 of file ecies.c.

References EC_KEY_free_d, EC_KEY_generate_key_d, ecies_key_alloc(), log_info, MEMORYBUF, and ssl_error_string().

Referenced by ecies_encrypt().

5.255.1.7 void ecies_key_free (EC_KEY * key)

Free an ECIES key pair.

Returns:

This function returns no value.

Definition at line 48 of file ecies.c.

References EC_KEY_free_d.

5.255.1.8 EC_KEY* ecies_key_private (uint64_t format, placer_t data)

Definition at line 200 of file ecies.c.

References BN_bin2bn_d, BN_free_d, BN_hex2bn_d, EC_KEY_free_d, EC_KEY_set_private_key_d, ecies_key_alloc(), ECIES_PRIVATE_BINARY, ECIES_PRIVATE_HEX, log_info, MEMORYBUF, number, pl_char_get(), pl_data_get(), pl_length_get(), and ssl_error_string().

Referenced by ecies_decrypt().

5.255.1.9 `uchar_t* ecies_key_private_bin (EC_KEY * key, size_t * olen)`

Return an ECIES private key as binary data.

Parameters:

key the input ECIES key pair.

olen a pointer to store the length of the returned key.

Returns:

NULL on failure, or a pointer to the raw private key.

Definition at line 360 of file ecies.c.

References BN_bn2bin_d, BN_num_bytes_d, EC_KEY_get0_private_key_d, log_info, MEMORYBUF, mm_sec_alloc(), mm_sec_free(), and ssl_error_string().

5.255.1.10 `stringer_t* ecies_key_private_hex (EC_KEY * key)`

Return an ECIES private key as a hex string.

Parameters:

key the input ECIES key pair.

Returns:

NULL on failure, or the hex-formatted private key as a managed string.

Definition at line 326 of file ecies.c.

References BN_bn2hex_d, CONTIGUOUS, EC_KEY_get0_private_key_d, log_pedantic, MANAGED_T, MEMORYBUF, ns_length_get(), ns_wipe(), OPENSSL_free_d, SECURE, ssl_error_string(), and st_import_opts().

5.255.1.11 `EC_KEY* ecies_key_public (uint64_t format, placer_t data)`

Definition at line 132 of file ecies.c.

References EC_KEY_check_key_d, EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_set_public_key_d, EC_POINT_free_d, EC_POINT_hex2point_d, EC_POINT_new_d, EC_POINT_oct2point_d, ecies_key_alloc(), ECIES_PUBLIC_BINARY, ECIES_PUBLIC_HEX, log_info, MEMORYBUF, pl_char_get(), pl_data_get(), pl_length_get(), and ssl_error_string().

Referenced by ecies_decrypt(), and ecies_encrypt().

5.255.1.12 `uchar_t* ecies_key_public_bin (EC_KEY * key, size_t * olen)`

Return an ECIES public key as binary data.

Parameters:

key the input ECIES key pair.

olen a pointer to store the length of the returned key.

Returns:

NULL on failure, or a pointer to the raw public key.

Definition at line 289 of file ecies.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2oct_d, log_info, MEMORYBUF, mm_alloc(), mm_free(), and ssl_error_string().

5.255.1.13 **stringer_t* ecies_key_public_hex (EC_KEY * *key*)**

Return an ECIES public key as a null-terminated hex string.

Parameters:

key the input ECIES key pair.

Returns:

NULL on failure, or the hex-formatted public key as a null-terminated string.

Definition at line 256 of file ecies.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2hex_d, log_info, MEMORYBUF, ns_length_get(), OPENSSL_free_d, ssl_error_string(), and st_import().

5.255.1.14 **bool_t ecies_start (void)**

Definition at line 16 of file ecies.c.

References ECIES_CIPHER, ecies_cipher_evp, ECIES_CURVE, ecies_curve_group, ECIES_ENVELOPE, ecies_envelope_evp, ecies_group(), ECIES_HMAC, ecies_hmac_evp, EVP_get_cipherbyname_d, EVP_get_digestbyname_d, log_error, and OBJ_nid2sn_d.

5.255.1.15 **void ecies_stop (void)**

Destroy all initialized ECIES curve group information.

Returns:

This function returns no value.

Definition at line 31 of file ecies.c.

References EC_GROUP_free_d, and ecies_curve_group.

5.255.2 **Variable Documentation**

5.255.2.1 **const EVP_CIPHER* ecies_cipher_evp**

Definition at line 13 of file ecies.c.

Referenced by deprecated_ecies_start(), and ecies_start().

5.255.2.2 **EC_GROUP* ecies_curve_group**

Definition at line 10 of file ecies.c.

Referenced by deprecated_ecies_key_alloc(), deprecated_ecies_start(), deprecated_ecies_stop(), ecies_key_alloc(), ecies_start(), and ecies_stop().

5.255.2.3 **const EVP_MD* ecies_envelope_evp**

Definition at line 12 of file ecies.c.

Referenced by deprecated_ecies_envelope_derivation(), deprecated_ecies_start(), ecies_envelope_derivation(), and ecies_start().

5.255.2.4 `const EVP_MD* ecies_hmac_evp`

Definition at line 11 of file ecies.c.

Referenced by `deprecated_ecies_start()`, and `ecies_start()`.

5.256 src/providers/cryptography/hash.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * hash_digest](#) ([digest_t * digest](#), [stringer_t * s](#), [stringer_t * output](#))

[hash.c](#)

- [stringer_t * hash_md4](#) ([stringer_t * s](#), [stringer_t * output](#))
- [stringer_t * hash_md5](#) ([stringer_t * s](#), [stringer_t * output](#))
- [stringer_t * hash_sha](#) ([stringer_t * s](#), [stringer_t * output](#))
- [stringer_t * hash_sha1](#) ([stringer_t * s](#), [stringer_t * output](#))
- [stringer_t * hash_sha224](#) ([stringer_t * s](#), [stringer_t * output](#))
- [stringer_t * hash_sha256](#) ([stringer_t * s](#), [stringer_t * output](#))
- [stringer_t * hash_sha384](#) ([stringer_t * s](#), [stringer_t * output](#))
- [stringer_t * hash_sha512](#) ([stringer_t * s](#), [stringer_t * output](#))
- [stringer_t * hash_ripemd160](#) ([stringer_t * s](#), [stringer_t * output](#))

5.256.1 Function Documentation

5.256.1.1 stringer_t* hash_digest (digest_t * *digest*, stringer_t * *s*, stringer_t * *output*)

[hash.c](#)

Definition at line 10 of file [hash.c](#).

References [EVP_DigestFinal_d](#), [EVP_DigestInit_ex_d](#), [EVP_DigestUpdate_d](#), [EVP_MD_CTX_cleanup_d](#), [EVP_MD_CTX_init_d](#), [EVP_MD_size_d](#), [log_pedantic](#), [MEMORYBUF](#), [ssl_error_string\(\)](#), [st_alloc\(\)](#), [st_avail_get\(\)](#), [st_data_get\(\)](#), [st_empty](#), [st_free\(\)](#), [st_length_get\(\)](#), [st_length_set\(\)](#), [st_valid_avail\(\)](#), [st_valid_destination\(\)](#), and [st_valid_tracked\(\)](#).

Referenced by [hash_md4\(\)](#), [hash_md5\(\)](#), [hash_ripemd160\(\)](#), [hash_sha\(\)](#), [hash_sha1\(\)](#), [hash_sha224\(\)](#), [hash_sha256\(\)](#), [hash_sha384\(\)](#), and [hash_sha512\(\)](#).

5.256.1.2 stringer_t* hash_md4 (stringer_t * *s*, stringer_t * *output*)

Definition at line 86 of file [hash.c](#).

References [EVP_md4_d](#), and [hash_digest\(\)](#).

5.256.1.3 stringer_t* hash_md5 (stringer_t * *s*, stringer_t * *output*)

Definition at line 90 of file [hash.c](#).

References [EVP_md5_d](#), and [hash_digest\(\)](#).

5.256.1.4 stringer_t* hash_ripemd160 (stringer_t * *s*, stringer_t * *output*)

Definition at line 118 of file [hash.c](#).

References [EVP_ripemd160_d](#), and [hash_digest\(\)](#).

5.256.1.5 stringer_t* hash_sha (stringer_t * s, stringer_t * output)

Definition at line 94 of file hash.c.

References `EVP_sha_d`, and `hash_digest()`.

5.256.1.6 stringer_t* hash_sha1 (stringer_t * s, stringer_t * output)

Definition at line 98 of file hash.c.

References `EVP_sha1_d`, and `hash_digest()`.

Referenced by `deprecated_symmetric_key()`, and `symmetric_key()`.

5.256.1.7 stringer_t* hash_sha224 (stringer_t * s, stringer_t * output)

Definition at line 102 of file hash.c.

References `EVP_sha224_d`, and `hash_digest()`.

Referenced by `deprecated_symmetric_key()`, and `symmetric_key()`.

5.256.1.8 stringer_t* hash_sha256 (stringer_t * s, stringer_t * output)

Definition at line 106 of file hash.c.

References `EVP_sha256_d`, and `hash_digest()`.

Referenced by `deprecated_symmetric_key()`, and `symmetric_key()`.

5.256.1.9 stringer_t* hash_sha384 (stringer_t * s, stringer_t * output)

Definition at line 110 of file hash.c.

References `EVP_sha384_d`, and `hash_digest()`.

5.256.1.10 stringer_t* hash_sha512 (stringer_t * s, stringer_t * output)

Definition at line 114 of file hash.c.

References `EVP_sha512_d`, and `hash_digest()`.

Referenced by `auth_legacy()`, `encrypted_chunk_get()`, `encrypted_chunk_set()`, `org_signet_fingerprint()`, `signature_full_get()`, `signature_full_verify()`, `signature_tree_add()`, `signature_tree_get()`, `signature_tree_verify()`, and `user_signet_fingerprint()`.

5.257 src/providers/cryptography/hmac.c File Reference

```
#include "magma.h"
```

Functions

- `stringer_t * deprecated_hmac_digest (digest_t *digest, stringer_t *s, stringer_t *key, stringer_t *output)`
Performs an HMAC using the specified digest and key.
- `stringer_t * deprecated_hmac_md4 (stringer_t *s, stringer_t *key, stringer_t *output)`
- `stringer_t * deprecated_hmac_md5 (stringer_t *s, stringer_t *key, stringer_t *output)`
- `stringer_t * deprecated_hmac_sha (stringer_t *s, stringer_t *key, stringer_t *output)`
- `stringer_t * deprecated_hmac_sha1 (stringer_t *s, stringer_t *key, stringer_t *output)`
- `stringer_t * deprecated_hmac_sha224 (stringer_t *s, stringer_t *key, stringer_t *output)`
- `stringer_t * deprecated_hmac_sha256 (stringer_t *s, stringer_t *key, stringer_t *output)`
- `stringer_t * deprecated_hmac_sha384 (stringer_t *s, stringer_t *key, stringer_t *output)`
- `stringer_t * deprecated_hmac_sha512 (stringer_t *s, stringer_t *key, stringer_t *output)`
- `stringer_t * deprecated_hmac_ripemd160 (stringer_t *s, stringer_t *key, stringer_t *output)`

5.257.1 Function Documentation

5.257.1.1 `stringer_t* deprecated_hmac_digest (digest_t *digest, stringer_t *s, stringer_t *key, stringer_t *output)`

Performs an HMAC using the specified digest and key. hmac.c

Parameters:

digest Digest to be used with the HMAC.
s Input data.
key Key used in HMAC.
output Stringer containing buffer for output.

Returns:

The managed string containing the resulting HMAC or NULL if an error occurs.

Definition at line 20 of file hmac.c.

References `EVP_MD_size_d`, `HMAC_CTX_cleanup_d`, `HMAC_CTX_init_d`, `HMAC_Final_d`, `HMAC_Init_ex_d`, `HMAC_Update_d`, `log_pedantic`, `st_alloc()`, `st_avail_get()`, `st_data_get()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_set()`, `st_valid_avail()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `deprecated_hmac_md4()`, `deprecated_hmac_md5()`, `deprecated_hmac_ripemd160()`, `deprecated_hmac_sha()`, `deprecated_hmac_sha1()`, `deprecated_hmac_sha224()`, `deprecated_hmac_sha256()`, `deprecated_hmac_sha384()`, and `deprecated_hmac_sha512()`.

5.257.1.2 `stringer_t* deprecated_hmac_md4 (stringer_t *s, stringer_t *key, stringer_t *output)`

Definition at line 90 of file hmac.c.

References `deprecated_hmac_digest()`, and `EVP_md4_d`.

5.257.1.3 `stringer_t* deprecated_hmac_md5 (stringer_t *s, stringer_t *key, stringer_t *output)`

Definition at line 94 of file hmac.c.

References `deprecated_hmac_digest()`, and `EVP_md5_d`.

5.257.1.4 stringer_t* deprecated_hmac_ripemd160 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 122 of file hmac.c.

References deprecated_hmac_digest(), and EVP_ripemd160_d.

5.257.1.5 stringer_t* deprecated_hmac_sha (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 98 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha_d.

5.257.1.6 stringer_t* deprecated_hmac_sha1 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 102 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha1_d.

5.257.1.7 stringer_t* deprecated_hmac_sha224 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 106 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha224_d.

5.257.1.8 stringer_t* deprecated_hmac_sha256 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 110 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha256_d.

5.257.1.9 stringer_t* deprecated_hmac_sha384 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 114 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha384_d.

5.257.1.10 stringer_t* deprecated_hmac_sha512 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 118 of file hmac.c.

References deprecated_hmac_digest(), and EVP_sha512_d.

5.258 src/providers/deprecated/hmac.c File Reference

```
#include "magma.h"
#include "deprecated.h"
```

Functions

- [stringer_t * hmac_digest](#) ([digest_t * digest](#), [stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))

Performs an HMAC using the specified digest and key.

- [stringer_t * hmac_md4](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))
- [stringer_t * hmac_md5](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))
- [stringer_t * hmac_sha](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))
- [stringer_t * hmac_sha1](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))
- [stringer_t * hmac_sha224](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))
- [stringer_t * hmac_sha256](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))
- [stringer_t * hmac_sha384](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))
- [stringer_t * hmac_sha512](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))
- [stringer_t * hmac_ripemd160](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))

5.258.1 Function Documentation

5.258.1.1 [stringer_t* hmac_digest](#) ([digest_t * digest](#), [stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))

Performs an HMAC using the specified digest and key. hmac.c

Parameters:

- digest*** Digest to be used with the HMAC.
- s*** Input data.
- key*** Key used in HMAC.
- output*** Stringer containing buffer for output.

Returns:

The managed string containing the resulting HMAC or NULL if an error occurs.

Definition at line 21 of file hmac.c.

References [EVP_MD_size_d](#), [HMAC_CTX_cleanup_d](#), [HMAC_CTX_init_d](#), [HMAC_Final_d](#), [HMAC_Init_ex_d](#), [HMAC_Update_d](#), [log_pedantic](#), [st_alloc\(\)](#), [st_avail_get\(\)](#), [st_data_get\(\)](#), [st_empty](#), [st_free\(\)](#), [st_length_get\(\)](#), [st_length_set\(\)](#), [st_valid_avail\(\)](#), [st_valid_destination\(\)](#), and [st_valid_tracked\(\)](#).

Referenced by [hmac_md4\(\)](#), [hmac_md5\(\)](#), [hmac_ripemd160\(\)](#), [hmac_sha\(\)](#), [hmac_sha1\(\)](#), [hmac_sha224\(\)](#), [hmac_sha256\(\)](#), [hmac_sha384\(\)](#), and [hmac_sha512\(\)](#).

5.258.1.2 [stringer_t* hmac_md4](#) ([stringer_t * s](#), [stringer_t * key](#), [stringer_t * output](#))

Definition at line 91 of file hmac.c.

References [EVP_md4_d](#), and [hmac_digest\(\)](#).

5.258.1.3 stringer_t* hmac_md5 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 95 of file hmac.c.

References `EVP_md5_d`, and `hmac_digest()`.

5.258.1.4 stringer_t* hmac_ripemd160 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 123 of file hmac.c.

References `EVP_ripemd160_d`, and `hmac_digest()`.

5.258.1.5 stringer_t* hmac_sha (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 99 of file hmac.c.

References `EVP_sha_d`, and `hmac_digest()`.

5.258.1.6 stringer_t* hmac_sha1 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 103 of file hmac.c.

References `EVP_sha1_d`, and `hmac_digest()`.

5.258.1.7 stringer_t* hmac_sha224 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 107 of file hmac.c.

References `EVP_sha224_d`, and `hmac_digest()`.

5.258.1.8 stringer_t* hmac_sha256 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 111 of file hmac.c.

References `EVP_sha256_d`, and `hmac_digest()`.

5.258.1.9 stringer_t* hmac_sha384 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 115 of file hmac.c.

References `EVP_sha384_d`, and `hmac_digest()`.

5.258.1.10 stringer_t* hmac_sha512 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 119 of file hmac.c.

References `EVP_sha512_d`, and `hmac_digest()`.

5.259 src/providers/cryptography/openssl.c File Reference

```
#include "magma.h"
```

Functions

- `const char * lib_version_openssl (void)`
Return the version string of the OpenSSL library.
- `bool_t lib_load_openssl (void)`
Initialize the OpenSSL library and bind dynamically to the exported functions that are required.
- `bool_t ssl_start (void)`
Initialize the OpenSSL facility.
- `void ssl_stop (void)`
Stop the openssl library and cleanup all associated data structures.
- `void ssl_thread_stop (void)`
Free the calling thread's SSL error queue.
- `void ssl_locking_callback (int mode, int n, const char *file, int line)`
The locking function callback necessary for all multi-threaded applications using OpenSSL.
- `unsigned long ssl_thread_id_callback (void)`
The thread id callback necessary for all multi-threaded applications using OpenSSL.
- `EC_KEY * ssl_ecdh_exchange_callback (SSL *ssl, int is_export, int keylength)`
Callback handler for the EC Diffie-Hellman parameter generation process necessary for PFS.
- `char * ssl_error_string (chr_t *buffer, int_t length)`
Get a textual representation of the last OpenSSL error message.
- `bool_t ssl_verify_privkey (const char *keyfile)`
Verify that the filename contains a valid private key in PEM format.

Variables

- `char ssl_version [16]`
- `DH * dh2048`
- `DH * dh4096`
- `pthread_mutex_t ** ssl_locks = NULL`
- `pthread_mutex_t dhparam_lock`
- `EC_KEY * ecdh512 = NULL`
- `EC_KEY * ecdh1024 = NULL`

5.259.1 Function Documentation

5.259.1.1 `bool_t lib_load_openssl (void)`

Initialize the OpenSSL library and bind dynamically to the exported functions that are required. [openssl.c](#)

Returns:

true on success or false on failure.

Definition at line 29 of file openssl.c.

References `lib_symbols()`, `M_BIND`, `ns_length_get()`, `pl_char_get()`, `pl_length_int()`, `placer_t`, `ssl_version`, `SSL_version_str_d`, `symbol_t`, and `tok_get_ns()`.

Referenced by `lib_load()`.

5.259.1.2 `const char* lib_version_openssl (void)`

Return the version string of the OpenSSL library.

Returns:

a pointer to a character string containing the libopenssl version information.

Definition at line 20 of file openssl.c.

References `ssl_version`.

Referenced by `lib_load()`.

5.259.1.3 `EC_KEY* ssl_ecdh_exchange_callback (SSL * ssl, int is_export, int keylength)`

Callback handler for the EC Diffie-Hellman parameter generation process necessary for PFS.

Parameters:

ssl the SSL session for which the callback was triggered.

is_export around here all we export is freedom.

keylength the length, in bits, of the ECCH key to be generated.

Returns:

a pointer to a ECDH key of proper size with parameters generated, or NULL on failure.

Definition at line 259 of file openssl.c.

References `EC_GROUP_set_point_conversion_form_d`, `EC_KEY_get0_group_d`, `EC_KEY_new_by_curve_name_d`, `ecdh1024`, `ecdh512`, and `log_error`.

Referenced by `tls_server_create()`.

5.259.1.4 `char* ssl_error_string (chr_t * buffer, int_t length)`

Get a textual representation of the last OpenSSL error message.

Parameters:

buffer a buffer that will receive the last OpenSSL error message.

length the size, in bytes, of the buffer that will contain the last OpenSSL error message.

Returns:

NULL on failure, or a pointer to the buffer where the last OpenSSL error message has been stored.

Definition at line 294 of file openssl.c.

References ERR_error_string_n_d, ERR_get_error_d, and log_pedantic.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_ecies_group(), deprecated_ecies_key_alloc(), deprecated_ecies_key_create(), deprecated_ecies_key_private(), deprecated_ecies_key_private_bin(), deprecated_ecies_key_private_hex(), deprecated_ecies_key_public(), deprecated_ecies_key_public_bin(), deprecated_ecies_key_public_hex(), deprecated_symmetric_decrypt(), dh_exchange_2048(), dh_exchange_4096(), dh_params_generate(), digest_id(), digest_length_output(), digest_name(), ecies_decrypt(), ecies_encrypt(), ecies_group(), ecies_key_alloc(), ecies_key_create(), ecies_key_private(), ecies_key_private_bin(), ecies_key_private_hex(), ecies_key_public(), ecies_key_public_bin(), ecies_key_public_hex(), ed25519_sign(), ed25519_verify(), hash_digest(), prime_start(), rand_get_int16(), rand_get_int32(), rand_get_int64(), rand_get_int8(), rand_get_uint16(), rand_get_uint32(), rand_get_uint64(), rand_get_uint8(), secp256k1_alloc(), secp256k1_compute_kek(), secp256k1_generate(), secp256k1_private_get(), secp256k1_private_set(), secp256k1_public_get(), secp256k1_public_set(), ssl_verify_privkey(), stacie_decrypt(), stacie_derive_key(), stacie_derive_seed(), stacie_derive_token(), stacie_realm_key(), symmetric_decrypt(), tls_client_alloc(), and tls_server_alloc().

5.259.1.5 void ssl_locking_callback (int mode, int n, const char *file, int line)

The locking function callback necessary for all multi-threaded applications using OpenSSL.

See also:

CRYPTO_set_locking_callback()

Parameters:

mode a bitmask specifying the requested openssl locking operation (CRYPTO_LOCK, CRYPTO_WRITE, CRYPTO_READ, or CRYPTO_UNLOCK).

n the zero-based index of the lock that is the target of the requested operation.

file a null-terminated string containing the filename of the function setting the lock, for debugging purposes.

line the line number of the function setting the lock, for debugging purposes.

Returns:

This function returns no value.

Definition at line 222 of file openssl.c.

References mutex_lock(), mutex_unlock(), and ssl_locks.

Referenced by ssl_start().

5.259.1.6 bool_t ssl_start (void)

Initialize the OpenSSL facility.

Note:

First, this function initializes the mutexes necessary for the locking function callback that openssl uses for shared data structures in multi-threaded applications. Next, the DKIM key is retrieved if the [magma.dkim.enabled](#) configuration variable is set.

Returns:

true if openssl was initialized successfully, or false on failure.

Definition at line 114 of file openssl.c.

References CRYPTO_num_locks_d, CRYPTO_set_id_callback_d, CRYPTO_set_locked_mem_functions_d, CRYPTO_set_locking_callback_d, dhparam_lock, log_critical, mm_alloc(), mm_sec_alloc(), mm_sec_free(), mutex_init(), OPENSSL_add_all_algorithms_noconf_d, SSL_library_init_d, SSL_load_error_strings_d, ssl_locking_callback(), ssl_locks, and ssl_thread_id_callback().

Referenced by process_start().

5.259.1.7 void ssl_stop (void)

Stop the openssl library and cleanup all associated data structures.

Returns:

This function returns no values.

Definition at line 164 of file openssl.c.

References ASN1_STRING_TABLE_cleanup_d, BIO_sock_cleanup_d, COMP_zlib_cleanup_d, CONF_modules_unload_d, CRYPTO_cleanup_all_ex_data_d, CRYPTO_num_locks_d, CRYPTO_set_id_callback_d, CRYPTO_set_locking_callback_d, dhparam_lock, EC_KEY_free_d, ecdh1024, ecdh512, ENGINE_cleanup_d, ERR_free_strings_d, ERR_remove_thread_state_d, EVP_cleanup_d, mm_free(), mutex_destroy(), OBJ_cleanup_d, OBJ_NAME_cleanup_d, sk_pop_free_d, and ssl_locks.

Referenced by process_stop().

5.259.1.8 unsigned long ssl_thread_id_callback (void)

The thread id callback necessary for all multi-threaded applications using OpenSSL.

See also:

CRYPTO_set_id_callback()

Returns:

the id of the calling thread.

Definition at line 247 of file openssl.c.

References thread_get_thread_id().

Referenced by ssl_start().

5.259.1.9 void ssl_thread_stop (void)

Free the calling thread's SSL error queue.

See also:

[ssl_thread_stop\(\)](#)

Returns:

This function returns no value.

Definition at line 208 of file openssl.c.

References ERR_remove_thread_state_d.

Referenced by thread_stop().

5.259.1.10 `bool_t ssl_verify_privkey (const char * keyfile)`

Verify that the filename contains a valid private key in PEM format.

Parameters:

keyfile the pathname of the private key.

Returns:

true if the the key is valid, or false if it is not.

Definition at line 314 of file openssl.c.

References `log_pedantic`, `MEMORYBUF`, `SSL_CTX_free_d`, `SSL_CTX_new_d`, `SSL_CTX_use_PrivateKey_file_d`, `ssl_error_string()`, and `SSLv23_client_method_d`.

Referenced by `dkim_start()`.

5.259.2 Variable Documentation

5.259.2.1 `DH* dh2048`

Definition at line 11 of file parameters.c.

Referenced by `dh_exchange_2048()`, and `dh_params_generate()`.

5.259.2.2 `DH * dh4096`

Definition at line 11 of file parameters.c.

Referenced by `dh_exchange_4096()`, and `dh_params_generate()`.

5.259.2.3 `pthread_mutex_t dhparam_lock`

Definition at line 10 of file parameters.c.

Referenced by `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_params_generate()`, `ssl_start()`, and `ssl_stop()`.

5.259.2.4 `EC_KEY * ecdh1024 = NULL`

Definition at line 14 of file openssl.c.

Referenced by `ssl_ecdh_exchange_callback()`, and `ssl_stop()`.

5.259.2.5 `EC_KEY* ecdh512 = NULL`

Definition at line 14 of file openssl.c.

Referenced by `ssl_ecdh_exchange_callback()`, and `ssl_stop()`.

5.259.2.6 `pthread_mutex_t** ssl_locks = NULL`

Definition at line 12 of file openssl.c.

Referenced by `ssl_locking_callback()`, `ssl_start()`, and `ssl_stop()`.

5.259.2.7 char ssl_version[16]

Definition at line 10 of file openssl.c.

Referenced by lib_load_openssl(), and lib_version_openssl().

5.260 src/providers/cryptography/parameters.c File Reference

```
#include "magma.h"
```

Functions

- int [dh_params_generate_callback](#) (int p, int n, BN_GENCB *cb)

The DH param generation callback.

- void [dh_params_generate](#) (void)

TODO: Spawn a thread at startup to generate the keys, and block the dh_exchange functions below from returning until its done.

- DH * [dh_params_2048](#) (void)
- DH * [dh_params_4096](#) (void)
- DH * [dh_static_2048](#) (SSL *ssl, int is_export, int keylength)
- DH * [dh_static_4096](#) (SSL *ssl, int is_export, int keylength)
- DH * [dh_exchange_2048](#) (SSL *ssl, int is_export, int keylength)

Callback handler for the Diffie-Hellman parameter generation process necessary for PFS.

- DH * [dh_exchange_4096](#) (SSL *ssl, int is_export, int keylength)

Callback handler for the Diffie-Hellman parameter generation process necessary for PFS.

Variables

- pthread_mutex_t [dhparam_lock](#)
- DH * [dh2048](#) = NULL
- DH * [dh4096](#) = NULL

5.260.1 Function Documentation

5.260.1.1 DH* [dh_exchange_2048](#) (SSL *ssl, int is_export, int keylength)

Callback handler for the Diffie-Hellman parameter generation process necessary for PFS. [parameters.c](#)

Parameters:

ssl the SSL session for which the callback was triggered.

is_export We ignore this parameter, as we don't support insecure cipher suites.

keylength the length, in bits, of the DH key to be generated is also ignored.

Returns:

a pointer to a Diffie-Hellman structure with a 2048 bit prime which is used for the session key, or NULL on failure.

Definition at line 270 of file parameters.c.

References [dh2048](#), [DH_check_d](#), [DH_free_d](#), [DH_generate_parameters_ex_d](#), [DH_new_d](#), [dh_params_generate_callback\(\)](#), [dhparam_lock](#), [log_error](#), [log_pedantic](#), [MEMORYBUF](#), [mutex_lock\(\)](#), [mutex_unlock\(\)](#), [ssl_error_string\(\)](#), and [status](#).

Referenced by [tls_server_create\(\)](#).

5.260.1.2 DH* dh_exchange_4096 (SSL * ssl, int is_export, int keylength)

Callback handler for the Diffie-Hellman parameter generation process necessary for PFS.

Parameters:

ssl the SSL session for which the callback was triggered.

is_export We ignore this parameter, as we don't support insecure cipher suites.

keylength the length, in bits, of the DH key to be generated is also ignored.

Returns:

a pointer to a Diffie-Hellman structure with a 4096 bit prime which is used for the session key, or NULL on failure.

Definition at line 338 of file parameters.c.

References dh4096, DH_check_d, DH_free_d, DH_generate_parameters_ex_d, DH_new_d, dh_params_generate_callback(), dhparam_lock, log_error, log_pedantic, MEMORYBUF, mutex_lock(), mutex_unlock(), ssl_error_string(), and status.

Referenced by tls_server_create().

5.260.1.3 DH* dh_params_2048 (void)

Definition at line 140 of file parameters.c.

References BN_bin2bn_d, DH_free_d, and DH_new_d.

Referenced by dh_params_generate(), and dh_static_2048().

5.260.1.4 DH* dh_params_4096 (void)

Definition at line 187 of file parameters.c.

References BN_bin2bn_d, DH_free_d, and DH_new_d.

Referenced by dh_params_generate(), and dh_static_4096().

5.260.1.5 void dh_params_generate (void)

TODO: Spawn a thread at startup to generate the keys, and block the dh_exchange functions below from returning until its done.

Definition at line 25 of file parameters.c.

References magma_t::cryptography, dh2048, dh4096, DH_check_d, DH_free_d, DH_generate_parameters_ex_d, DH_new_d, dh_params_2048(), dh_params_4096(), dh_params_generate_callback(), dhparam_lock, magma_t::iface, log_error, log_pedantic, magma, MEMORYBUF, mutex_lock(), mutex_unlock(), ssl_error_string(), and status.

5.260.1.6 int dh_params_generate_callback (int p, int n, BN_GENCB * cb)

The DH param generation callback.

Definition at line 16 of file parameters.c.

References status.

Referenced by dh_exchange_2048(), dh_exchange_4096(), and dh_params_generate().

5.260.1.7 DH* dh_static_2048 (SSL * ssl, int is_export, int keylength)

Definition at line 255 of file parameters.c.

References dh_params_2048().

Referenced by tls_server_create().

5.260.1.8 DH* dh_static_4096 (SSL * ssl, int is_export, int keylength)

Definition at line 259 of file parameters.c.

References dh_params_4096().

Referenced by tls_server_create().

5.260.2 Variable Documentation**5.260.2.1 DH* dh2048 = NULL**

Definition at line 11 of file parameters.c.

Referenced by dh_exchange_2048(), and dh_params_generate().

5.260.2.2 DH * dh4096 = NULL

Definition at line 11 of file parameters.c.

Referenced by dh_exchange_4096(), and dh_params_generate().

5.260.2.3 pthread_mutex_t dhparam_lock

Definition at line 10 of file parameters.c.

Referenced by dh_exchange_2048(), dh_exchange_4096(), dh_params_generate(), ssl_start(), and ssl_stop().

5.261 src/providers/cryptography/random.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t](#) * [rand_choices](#) ([chr_t](#) *choices, [size_t](#) len, [stringer_t](#) *output)
Get a random string of data of a specified size, populated with characters from a chosen set.
- [size_t](#) [rand_write](#) ([stringer_t](#) *s)
Fill a managed string with random data.
- [uint64_t](#) [rand_get_uint64](#) (void)
Generate a random unsigned 64 bit number.
- [uint32_t](#) [rand_get_uint32](#) (void)
Generate a random unsigned 32 bit number.
- [uint16_t](#) [rand_get_uint16](#) (void)
Generate a random unsigned 16 bit number.
- [uint8_t](#) [rand_get_uint8](#) (void)
Generate a random unsigned 8 bit number.
- [int64_t](#) [rand_get_int64](#) (void)
Generate a random signed 64 bit number.
- [int32_t](#) [rand_get_int32](#) (void)
- [int16_t](#) [rand_get_int16](#) (void)
- [int8_t](#) [rand_get_int8](#) (void)
- [bool_t](#) [rand_thread_start](#) (void)
- [bool_t](#) [rand_start](#) (void)
Initialize random number generation services and seed the generator.
- void [rand_stop](#) (void)

Variables

- __thread [uint_t](#) [rand_ctx](#) = 0

5.261.1 Function Documentation

5.261.1.1 [stringer_t](#)* [rand_choices](#) ([chr_t](#) * choices, [size_t](#) len, [stringer_t](#) * output)

Get a random string of data of a specified size, populated with characters from a chosen set.

Parameters:

- choices** a pointer to a null-terminated string containing a pool of characters from which the contents of the random data will be selected.
- len** the length, in bytes, of the random string that will be returned to the caller.

Returns:

NULL on failure or a pointer to a managed string containing len bytes of random data on success.

Definition at line 23 of file random.c.

References `log_pedantic`, `ns_length_get()`, `RAND_bytes_d`, `st_alloc()`, `st_avail_get()`, `st_cleanup`, `st_data_get()`, `st_length_set()`, `st_opt_get()`, `st_valid_destination()`, and `st_valid_tracked()`.

Referenced by `mail_create_message()`, `register_print_captcha()`, `register_session_generate()`, `smtp_bounce()`, and `smtp_reply()`.

5.261.1.2 `int16_t rand_get_int16 (void)`

Definition at line 248 of file random.c.

References `buflen`, `bufptr`, `log_pedantic`, `RAND_bytes_d`, `rand_ctx`, and `ssl_error_string()`.

5.261.1.3 `int32_t rand_get_int32 (void)`

Definition at line 227 of file random.c.

References `buflen`, `bufptr`, `log_pedantic`, `RAND_bytes_d`, `rand_ctx`, and `ssl_error_string()`.

5.261.1.4 `int64_t rand_get_int64 (void)`

Generate a random signed 64 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated signed 64 bit integer.

See also:

`RAND_bytes()`

Definition at line 204 of file random.c.

References `buflen`, `bufptr`, `log_pedantic`, `RAND_bytes_d`, `rand_ctx`, and `ssl_error_string()`.

Referenced by `portal_message_source()`.

5.261.1.5 `int8_t rand_get_int8 (void)`

Definition at line 269 of file random.c.

References `buflen`, `bufptr`, `log_pedantic`, `RAND_bytes_d`, `rand_ctx`, and `ssl_error_string()`.

5.261.1.6 `uint16_t rand_get_uint16 (void)`

Generate a random unsigned 16 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated unsigned 16 bit integer.

See also:

RAND_bytes()

Definition at line 159 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

5.261.1.7 uint32_t rand_get_uint32 (void)

Generate a random unsigned 32 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated unsigned 32 bit integer.

See also:

RAND_bytes()

Definition at line 137 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

Referenced by auth_login(), process_maint(), register_captcha_generate(), register_captcha_random_font(), register_captcha_write_noise(), and smtp_client_connect().

5.261.1.8 uint64_t rand_get_uint64 (void)

Generate a random unsigned 64 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated unsigned 64 bit integer.

See also:

RAND_bytes()

Definition at line 113 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

Referenced by mail_mime_generate_boundary(), sess_key(), smtp_store_spamsig(), and spool_mktemp().

5.261.1.9 uint8_t rand_get_uint8 (void)

Generate a random unsigned 8 bit number.

Note:

This function attempts to generate random data securely, but falls back on the pseudo-random number generator.

Returns:

the newly generated unsigned 8 bit integer.

See also:

RAND_bytes()

Definition at line 181 of file random.c.

References buflen, bufptr, log_pedantic, RAND_bytes_d, rand_ctx, and ssl_error_string().

5.261.1.10 bool_t rand_start (void)

Initialize random number generation services and seed the generator. [random.c](#)

Note:

The default seed source for cryptographically secure generation routines is the system device /dev/random.

Returns:

false on failure, true on success.

Definition at line 308 of file random.c.

References magma_t::cryptography, magma_t::iface, magma, rand_ctx, RAND_load_file_d, RAND_status_d, and thread_get_thread_id().

Referenced by process_start().

5.261.1.11 void rand_stop (void)

Definition at line 331 of file random.c.

References RAND_cleanup_d.

Referenced by process_stop().

5.261.1.12 bool_t rand_thread_start (void)

Definition at line 292 of file random.c.

References magma_t::cryptography, magma_t::iface, magma, rand_ctx, and thread_get_thread_id().

5.261.1.13 size_t rand_write (stringer_t * s)

Fill a managed string with random data.

Note:

This function generates random data using the cryptographically strong OpenSSL function RAND_bytes().

Parameters:

s the input managed string.

Returns:

0 on failure, or the total number of bytes written to the managed string.

See also:

RAND_bytes()

Definition at line 76 of file random.c.

References log_pedantic, RAND_bytes_d, st_avail_get(), st_data_get(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), and st_valid_tracked().

Referenced by aes_artifact_encrypt(), aes_chunk_encrypt(), deprecated_symmetric_vector(), encrypted_chunk_set(), signature_full_get(), signature_tree_get(), stacie_create_nonce(), stacie_create_salt(), stacie_create_shard(), stacie_encrypt(), and symmetric_vector().

5.261.2 Variable Documentation

5.261.2.1 __thread uint_t rand_ctx = 0

Definition at line 15 of file random.c.

Referenced by rand_get_int16(), rand_get_int32(), rand_get_int64(), rand_get_int8(), rand_get_uint16(), rand_get_uint32(), rand_get_uint64(), rand_get_uint8(), rand_start(), and rand_thread_start().

5.262 src/providers/cryptography/scramble.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t deprecated_scramble_total_length (scramble_t *buffer)`
Get the total length of a scrambled object.
- `uint64_t deprecated_scramble_body_hash (scramble_t *buffer)`
Get a hash of a scrambled object's (encrypted) body data.
- `uint64_t deprecated_scramble_orig_length (scramble_t *buffer)`
Get the length of the original (unencrypted) data underlying a scrambled object.
- `uint64_t deprecated_scramble_body_length (scramble_t *buffer)`
Get the length of a scrambled object's (encrypted) body data.
- `uint64_t deprecated_scramble_vector_length (scramble_t *buffer)`
Get the length of a scrambled object's IV.
- `void * deprecated_scramble_body_data (scramble_t *buffer)`
Get a pointer to the start of the encrypted data from a scrambled data header.
- `void * deprecated_scramble_vector_data (scramble_t *buffer)`
Get a pointer to the start of the IV from a scrambled data header.
- `scramble_t * deprecated_scramble_import (stringer_t *s)`
Return a managed string as a scrambled buffer, after validation.
- `scramble_t * deprecated_scramble_alloc (size_t length)`
Allocate a new scrambled data block.
- `void deprecated_scramble_free (scramble_t *buffer)`
Free a scrambled data block.
- `void deprecated_scramble_cleanup (scramble_t *buffer)`
Performed a checked free of a scramble buffer.
- `scramble_t * deprecated_scramble_encrypt (stringer_t *key, stringer_t *input)`
Scramble a block of data using an encryption key.
- `stringer_t * deprecated_scramble_decrypt (stringer_t *key, scramble_t *input)`
Un-scramble a block of data using an decryption key.

5.262.1 Function Documentation

5.262.1.1 `scramble_t * deprecated_scramble_alloc (size_t length)`

Allocate a new scrambled data block. `scramble.c`

Parameters:

length the length, in bytes, of the scrambled data buffer (should include the IV and encrypted body length).

Returns:

a pointer to a newly allocated scrambled data header.

Definition at line 143 of file scramble.c.

References mm_alloc(), and scramble_head_t.

Referenced by deprecated_scramble_encrypt().

5.262.1.2 void* deprecated_scramble_body_data (scramble_t * *buffer*)

Get a pointer to the start of the encrypted data from a scrambled data header.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

a pointer to the start of the scrambled data header's associated encrypted data body.

Definition at line 81 of file scramble.c.

References deprecated_scramble_vector_length(), and scramble_head_t.

Referenced by deprecated_scramble_decrypt(), deprecated_scramble_encrypt(), and deprecated_scramble_import().

5.262.1.3 uint64_t deprecated_scramble_body_hash (scramble_t * *buffer*)

Get a hash of a scrambled object's (encrypted) body data.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the 64-bit hash of the specified scrambled object's body data.

Definition at line 33 of file scramble.c.

References scramble_head_t.

5.262.1.4 uint64_t deprecated_scramble_body_length (scramble_t * *buffer*)

Get the length of a scrambled object's (encrypted) body data.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's body data.

Definition at line 57 of file scramble.c.

References scramble_head_t.

Referenced by deprecated_scramble_decrypt(), and deprecated_scramble_total_length().

5.262.1.5 void deprecated_scramble_cleanup (scramble_t * *buffer*)

Performed a checked free of a scramble buffer.

See also:

[scramble_free](#)

Parameters:

block the scramble buffer to be freed.

Returns:

This function returns no value.

Definition at line 166 of file scramble.c.

References mm_free().

5.262.1.6 stringer_t* deprecated_scramble_decrypt (stringer_t * *key*, scramble_t * *input*)

Un-scramble a block of data using an decryption key.

Parameters:

key a managed string containing the symmetric decryption key.

input a pointer to the scrambled data header to be decrypted.

Returns:

NULL on failure, or a managed string containing the verified decrypted data.

Definition at line 239 of file scramble.c.

References cipher_id(), deprecated_scramble_body_data(), deprecated_scramble_body_length(), deprecated_scramble_vector_data(), deprecated_scramble_vector_length(), deprecated_symmetric_decrypt(), deprecated_symmetric_key(), hash_adler32(), log_info, log_pedantic, MANAGEDBUF, PLACER, scramble_head_t, st_free(), and st_length_get().

Referenced by sess_get().

5.262.1.7 scramble_t* deprecated_scramble_encrypt (stringer_t * *key*, stringer_t * *input*)

Scramble a block of data using an encryption key.

Note:

This function is configured to use AES 256 in CBC mode.

Parameters:

key a managed string containing the symmetric encryption key.

input a managed string containing the data to be encrypted.

Returns:

NULL on failure, or a pointer to a scrambled data header containing the encrypted data and its metadata.

Definition at line 182 of file scramble.c.

References cipher_id(), cipher_numeric_id(), deprecated_scramble_alloc(), deprecated_scramble_body_data(), deprecated_scramble_vector_data(), deprecated_symmetric_encrypt(), deprecated_symmetric_key(), deprecated_symmetric_vector(), hash_adler32(), log_pedantic, MANAGEDBUF, mm_copy(), scramble_head_t, st_data_get(), st_free(), and st_length_get().

Referenced by sess_token().

5.262.1.8 void deprecated_scramble_free (scramble_t * *buffer*)

Free a scrambled data block.

Parameters:

buffer a pointer to the scrambled data header to be freed.

Returns:

This function returns no value.

Definition at line 153 of file scramble.c.

References mm_free().

Referenced by sess_token().

5.262.1.9 scramble_t* deprecated_scramble_import (stringer_t * *s*)

Return a managed string as a scrambled buffer, after validation.

Note:

The returned pointer is inside the existing stringer, so it doesn't need be freed.

Parameters:

s a managed string containing the serialized scrambled data.

Returns:

NULL on failure, or a pointer to the scrambled data header on success.

Definition at line 102 of file scramble.c.

References deprecated_scramble_body_data(), hash_adler32(), log_pedantic, scramble_head_t, st_data_get(), st_empty, and st_length_get().

Referenced by sess_get().

5.262.1.10 uint64_t deprecated_scramble_orig_length (scramble_t * *buffer*)

Get the length of the original (unencrypted) data underlying a scrambled object.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's original data.

Definition at line 45 of file scramble.c.

References scramble_head_t.

5.262.1.11 uint64_t deprecated_scramble_total_length (scramble_t * *buffer*)

Get the total length of a scrambled object. TODO: Create a scramble_export() function, and/or alter the import interfaces to make using managed strings easier. ie. the import/export functions could provide wrappers around ddecrypt/encrypt which return strings instead of scramble objects. HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the total length, in bytes, of the scrambled object.

Definition at line 19 of file scramble.c.

References deprecated_scramble_body_length(), deprecated_scramble_vector_length(), and scramble_head_t.

Referenced by sess_token().

5.262.1.12 void* deprecated_scramble_vector_data (scramble_t * *buffer*)

Get a pointer to the start of the IV from a scrambled data header.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

a pointer to the start of the scrambled data header's associated IV block.

Definition at line 91 of file scramble.c.

References scramble_head_t.

Referenced by deprecated_scramble_decrypt(), and deprecated_scramble_encrypt().

5.262.1.13 uint64_t deprecated_scramble_vector_length (scramble_t * *buffer*)

Get the length of a scrambled object's IV.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's IV.

Definition at line 69 of file scramble.c.

References scramble_head_t.

Referenced by deprecated_scramble_body_data(), deprecated_scramble_decrypt(), and deprecated_scramble_total_length().

5.263 src/providers/deprecated/scramble.c File Reference

```
#include "magma.h"
#include "deprecated.h"
```

Functions

- `uint64_t scramble_total_length (scramble_t *buffer)`
Get the total length of a scrambled object.
- `uint64_t scramble_body_hash (scramble_t *buffer)`
Get a hash of a scrambled object's (encrypted) body data.
- `uint64_t scramble_orig_length (scramble_t *buffer)`
Get the length of the original (unencrypted) data underlying a scrambled object.
- `uint64_t scramble_body_length (scramble_t *buffer)`
Get the length of a scrambled object's (encrypted) body data.
- `uint64_t scramble_vector_length (scramble_t *buffer)`
Get the length of a scrambled object's IV.
- `void * scramble_body_data (scramble_t *buffer)`
Get a pointer to the start of the encrypted data from a scrambled data header.
- `void * scramble_vector_data (scramble_t *buffer)`
Get a pointer to the start of the IV from a scrambled data header.
- `scramble_t * scramble_import (stringer_t *s)`
Return a managed string as a scrambled buffer, after validation.
- `scramble_t * scramble_alloc (size_t length)`
Allocate a new scrambled data block.
- `void scramble_free (scramble_t *buffer)`
Free a scrambled data block.
- `void scramble_cleanup (scramble_t *buffer)`
Performed a checked free of a scramble buffer.
- `scramble_t * scramble_encrypt (stringer_t *key, stringer_t *input)`
Scramble a block of data using an encryption key.
- `stringer_t * scramble_decrypt (stringer_t *key, scramble_t *input)`
Un-scramble a block of data using an decryption key.

5.263.1 Function Documentation

5.263.1.1 `scramble_t* scramble_alloc (size_t length)`

Allocate a new scrambled data block. `scramble.c`

Parameters:

length the length, in bytes, of the scrambled data buffer (should include the IV and encrypted body length).

Returns:

a pointer to a newly allocated scrambled data header.

Definition at line 143 of file scramble.c.

References mm_alloc(), and scramble_head_t.

Referenced by scramble_encrypt().

5.263.1.2 void* scramble_body_data (scramble_t * *buffer*)

Get a pointer to the start of the encrypted data from a scrambled data header.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

a pointer to the start of the scrambled data header's associated encrypted data body.

Definition at line 82 of file scramble.c.

References scramble_head_t, and scramble_vector_length().

Referenced by scramble_decrypt(), scramble_encrypt(), and scramble_import().

5.263.1.3 uint64_t scramble_body_hash (scramble_t * *buffer*)

Get a hash of a scrambled object's (encrypted) body data.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the 64-bit hash of the specified scrambled object's body data.

Definition at line 34 of file scramble.c.

References scramble_head_t.

5.263.1.4 uint64_t scramble_body_length (scramble_t * *buffer*)

Get the length of a scrambled object's (encrypted) body data.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's body data.

Definition at line 58 of file scramble.c.

References scramble_head_t.

Referenced by scramble_decrypt(), and scramble_total_length().

5.263.1.5 void scramble_cleanup (scramble_t * *buffer*)

Performed a checked free of a scramble buffer.

See also:

[scramble_free](#)

Parameters:

block the scramble buffer to be freed.

Returns:

This function returns no value.

Definition at line 166 of file scramble.c.

References `mm_free()`.

5.263.1.6 stringer_t* scramble_decrypt (stringer_t * *key*, scramble_t * *input*)

Un-scramble a block of data using an decryption key.

Parameters:

key a managed string containing the symmetric decryption key.

input a pointer to the scrambled data header to be decrypted.

Returns:

NULL on failure, or a managed string containing the verified decrypted data.

Definition at line 239 of file scramble.c.

References `cipher_id()`, `hash_adler32()`, `log_info`, `log_pedantic`, `MANAGEDBUF`, `PLACER`, `scramble_body_data()`, `scramble_body_length()`, `scramble_head_t`, `scramble_vector_data()`, `scramble_vector_length()`, `st_free()`, `st_length_get()`, `symmetric_decrypt()`, and `symmetric_key()`.

5.263.1.7 scramble_t* scramble_encrypt (stringer_t * *key*, stringer_t * *input*)

Scramble a block of data using an encryption key.

Note:

This function is configured to use AES 256 in CBC mode.

Parameters:

key a managed string containing the symmetric encryption key.

input a managed string containing the data to be encrypted.

Returns:

NULL on failure, or a pointer to a scrambled data header containing the encrypted data and its metadata.

Definition at line 182 of file scramble.c.

References `cipher_id()`, `cipher_numeric_id()`, `hash_adler32()`, `log_pedantic`, `MANAGEDBUF`, `mm_copy()`, `scramble_alloc()`, `scramble_body_data()`, `scramble_head_t`, `scramble_vector_data()`, `st_data_get()`, `st_free()`, `st_length_get()`, `symmetric_encrypt()`, `symmetric_key()`, and `symmetric_vector()`.

5.263.1.8 void scramble_free (scramble_t * *buffer*)

Free a scrambled data block.

Parameters:

buffer a pointer to the scrambled data header to be freed.

Returns:

This function returns no value.

Definition at line 153 of file scramble.c.

References mm_free().

5.263.1.9 scramble_t* scramble_import (stringer_t * *s*)

Return a managed string as a scrambled buffer, after validation.

Note:

The returned pointer is inside the existing stringer, so it doesn't need be freed.

Parameters:

s a managed string containing the serialized scrambled data.

Returns:

NULL on failure, or a pointer to the scrambled data header on success.

Definition at line 103 of file scramble.c.

References hash_adler32(), log_pedantic, scramble_body_data(), scramble_head_t, st_data_get(), st_empty, and st_length_get().

5.263.1.10 uint64_t scramble_orig_length (scramble_t * *buffer*)

Get the length of the original (unencrypted) data underlying a scrambled object.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's original data.

Definition at line 46 of file scramble.c.

References scramble_head_t.

5.263.1.11 uint64_t scramble_total_length (scramble_t * *buffer*)

Get the total length of a scrambled object. TODO: Create a scramble_export() function, and/or alter the import interfaces to make using managed strings easier. ie. the import/export functions could provide wrappers around ddecrypt/encrypt which return strings instead of scramble objects. HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the total length, in bytes, of the scrambled object.

Definition at line 20 of file scramble.c.

References `scramble_body_length()`, `scramble_head_t`, and `scramble_vector_length()`.

5.263.1.12 void* scramble_vector_data (scramble_t * *buffer*)

Get a pointer to the start of the IV from a scrambled data header.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

a pointer to the start of the scrambled data header's associated IV block.

Definition at line 92 of file scramble.c.

References `scramble_head_t`.

Referenced by `scramble_decrypt()`, and `scramble_encrypt()`.

5.263.1.13 uint64_t scramble_vector_length (scramble_t * *buffer*)

Get the length of a scrambled object's IV.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's IV.

Definition at line 70 of file scramble.c.

References `scramble_head_t`.

Referenced by `scramble_body_data()`, `scramble_decrypt()`, and `scramble_total_length()`.

5.264 src/providers/cryptography/symmetric.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * deprecated_symmetric_encrypt](#) ([cipher_t](#) *cipher, [stringer_t](#) *vector, [stringer_t](#) *key, [stringer_t](#) *input)
HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.
- [stringer_t * deprecated_symmetric_decrypt](#) ([cipher_t](#) *cipher, [stringer_t](#) *vector, [stringer_t](#) *key, [stringer_t](#) *input)
symmetric.c
- [stringer_t * deprecated_symmetric_vector](#) ([cipher_t](#) *cipher, [stringer_t](#) *output)
Generate an IV data suitable for the specified cipher.
- [stringer_t * deprecated_symmetric_key](#) ([cipher_t](#) *cipher, [stringer_t](#) *key, [stringer_t](#) *output)
Derive a symmetric key from a user's password.

5.264.1 Function Documentation

5.264.1.1 [stringer_t* deprecated_symmetric_decrypt](#) ([cipher_t](#) * *cipher*, [stringer_t](#) * *vector*, [stringer_t](#) * *key*, [stringer_t](#) * *input*)

symmetric.c

Definition at line 128 of file *symmetric.c*.

References [EVP_CIPHER_CTX_block_size_d](#), [EVP_CIPHER_CTX_cleanup_d](#), [EVP_CIPHER_CTX_ctrl_d](#), [EVP_CIPHER_CTX_init_d](#), [EVP_CIPHER_CTX_iv_length_d](#), [EVP_CIPHER_CTX_key_length_d](#), [EVP_CIPHER_flags_d](#), [EVP_DecryptFinal_ex_d](#), [EVP_DecryptInit_ex_d](#), [EVP_DecryptUpdate_d](#), [log_pedantic](#), [MEMORYBUF](#), [ssl_error_string\(\)](#), [st_alloc\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), and [st_length_set\(\)](#).

Referenced by [deprecated_scramble_decrypt\(\)](#).

5.264.1.2 [stringer_t* deprecated_symmetric_encrypt](#) ([cipher_t](#) * *cipher*, [stringer_t](#) * *vector*, [stringer_t](#) * *key*, [stringer_t](#) * *input*)

HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory. Encrypt a block of data using a symmetric cipher.

Parameters:

- cipher*** the cipher type being used to encrypt the data.
- vector*** the IV used for the encryption operation.
- key*** the symmetric encryption key used to encrypt the data.
- input*** a managed string containing the data to be encrypted.

Returns:

NULL on failure, or a pointer to a managed string containing the encrypted data.

Definition at line 20 of file *symmetric.c*.

References [EVP_CIPHER_CTX_block_size_d](#), [EVP_CIPHER_CTX_cleanup_d](#), [EVP_CIPHER_CTX_ctrl_d](#), [EVP_CIPHER_CTX_init_d](#), [EVP_CIPHER_CTX_iv_length_d](#), [EVP_CIPHER_CTX_key_length_d](#), [EVP_CIPHER_flags_d](#), [EVP_EncryptFinal_ex_d](#), [EVP_EncryptInit_ex_d](#), [EVP_EncryptUpdate_d](#), [log_pedantic](#), [st_alloc\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), and [st_length_set\(\)](#).

Referenced by [deprecated_scramble_encrypt\(\)](#).

5.264.1.3 stringer_t* deprecated_symmetric_key (cipher_t * *cipher*, stringer_t * *key*, stringer_t * *output*)

Derive a symmetric key from a user's password.

Note:

Depending on the length of the password, either SHA1, SHA224, or SHA256 may be used as the hashing algorithm. If output is passed as NULL, a new managed string will be allocated to hold the output of the operation.

Parameters:

cipher the specified cipher type.

key the user's password, as a managed string.

output the output managed string to receive the digested password. Can be NULL.

Returns:

NULL on failure, or a pointer to the managed string containing the key.

Definition at line 288 of file symmetric.c.

References cipher_key_length(), hash_sha1(), hash_sha224(), hash_sha256(), st_length_get(), st_length_set(), st_output(), and st_valid_tracked().

Referenced by deprecated_scramble_decrypt(), and deprecated_scramble_encrypt().

5.264.1.4 stringer_t* deprecated_symmetric_vector (cipher_t * *cipher*, stringer_t * *output*)

Generate an IV data suitable for the specified cipher.

Note:

If output is NULL, a new managed string will be allocated and returned by the function.

Parameters:

cipher the cipher type to be used.

output the managed string that will store the IV data.

Returns:

NULL on failure, or a pointer to the managed string that contains the IV data.

Definition at line 251 of file symmetric.c.

References cipher_vector_length(), log_pedantic, PLACER, rand_write(), st_data_get(), st_free(), st_length_set(), st_output(), and st_valid_tracked().

Referenced by deprecated_scramble_encrypt().

5.265 src/providers/deprecated/symmetric.c File Reference

```
#include "magma.h"
#include "deprecated.h"
```

Functions

- [stringer_t * symmetric_encrypt](#) ([cipher_t](#) *cipher, [stringer_t](#) *vector, [stringer_t](#) *key, [stringer_t](#) *input)
HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.
- [stringer_t * symmetric_decrypt](#) ([cipher_t](#) *cipher, [stringer_t](#) *vector, [stringer_t](#) *key, [stringer_t](#) *input)
symmetric.c
- [stringer_t * symmetric_vector](#) ([cipher_t](#) *cipher, [stringer_t](#) *output)
Generate an IV data suitable for the specified cipher.
- [stringer_t * symmetric_key](#) ([cipher_t](#) *cipher, [stringer_t](#) *key, [stringer_t](#) *output)
Derive a symmetric key from a user's password.

5.265.1 Function Documentation

5.265.1.1 stringer_t* symmetric_decrypt (cipher_t * cipher, stringer_t * vector, stringer_t * key, stringer_t * input)

symmetric.c

Definition at line 129 of file symmetric.c.

References [EVP_CIPHER_CTX_block_size_d](#), [EVP_CIPHER_CTX_cleanup_d](#), [EVP_CIPHER_CTX_ctrl_d](#), [EVP_CIPHER_CTX_init_d](#), [EVP_CIPHER_CTX_iv_length_d](#), [EVP_CIPHER_CTX_key_length_d](#), [EVP_CIPHER_flags_d](#), [EVP_DecryptFinal_ex_d](#), [EVP_DecryptInit_ex_d](#), [EVP_DecryptUpdate_d](#), [log_pedantic](#), [MEMORYBUF](#), [ssl_error_string\(\)](#), [st_alloc\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), and [st_length_set\(\)](#).

Referenced by [scramble_decrypt\(\)](#).

5.265.1.2 stringer_t* symmetric_encrypt (cipher_t * cipher, stringer_t * vector, stringer_t * key, stringer_t * input)

HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.
Encrypt a block of data using a symmetric cipher.

Parameters:

- cipher*** the cipher type being used to encrypt the data.
- vector*** the IV used for the encryption operation.
- key*** the symmetric encryption key used to encrypt the data.
- input*** a managed string containing the data to be encrypted.

Returns:

- NULL on failure, or a pointer to a managed string containing the encrypted data.

Definition at line 21 of file symmetric.c.

References [EVP_CIPHER_CTX_block_size_d](#), [EVP_CIPHER_CTX_cleanup_d](#), [EVP_CIPHER_CTX_ctrl_d](#), [EVP_CIPHER_CTX_init_d](#), [EVP_CIPHER_CTX_iv_length_d](#), [EVP_CIPHER_CTX_key_length_d](#), [EVP_CIPHER_flags_d](#), [EVP_EncryptFinal_ex_d](#), [EVP_EncryptInit_ex_d](#), [EVP_EncryptUpdate_d](#), [log_pedantic](#), [st_alloc\(\)](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), and [st_length_set\(\)](#).

Referenced by `scramble_encrypt()`.

5.265.1.3 `stringer_t* symmetric_key (cipher_t * cipher, stringer_t * key, stringer_t * output)`

Derive a symmetric key from a user's password.

Note:

Depending on the length of the password, either SHA1, SHA224, or SHA256 may be used as the hashing algorithm. If output is passed as NULL, a new managed string will be allocated to hold the output of the operation.

Parameters:

cipher the specified cipher type.

key the user's password, as a managed string.

output the output managed string to receive the digested password. Can be NULL.

Returns:

NULL on failure, or a pointer to the managed string containing the key.

Definition at line 289 of file `symmetric.c`.

References `cipher_key_length()`, `hash_sha1()`, `hash_sha224()`, `hash_sha256()`, `st_length_get()`, `st_length_set()`, `st_output()`, and `st_valid_tracked()`.

Referenced by `scramble_decrypt()`, and `scramble_encrypt()`.

5.265.1.4 `stringer_t* symmetric_vector (cipher_t * cipher, stringer_t * output)`

Generate an IV data suitable for the specified cipher.

Note:

If output is NULL, a new managed string will be allocated and returned by the function.

Parameters:

cipher the cipher type to be used.

output the managed string that will store the IV data.

Returns:

NULL on failure, or a pointer to the managed string that contains the IV data.

Definition at line 252 of file `symmetric.c`.

References `cipher_vector_length()`, `log_pedantic`, `PLACER`, `rand_write()`, `st_data_get()`, `st_free()`, `st_length_set()`, `st_output()`, and `st_valid_tracked()`.

Referenced by `scramble_encrypt()`.

5.266 src/providers/cryptography/tls.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t](#) [tls_server_create](#) (void *server, [uint_t](#) security_level)
Setup an TLS CTX for a server.
- void [tls_server_destroy](#) (void *server)
Destroy an TLS context associated with a server.
- [TLS](#) * [tls_server_alloc](#) (void *server, int sockd, int flags)
Create a TLS session for a file descriptor, and accept the client TLS/SSL handshake.
- void * [tls_client_alloc](#) ([int_t](#) sockd)
Establish an TLS client wrapper around a socket descriptor.
- void [tls_free](#) ([TLS](#) *tls)
Shutdown and free an TLS connection.
- [stringer_t](#) * [tls_cipher](#) ([TLS](#) *tls, [stringer_t](#) *output)
Return the name of the TLS cipher suite assoicated with a connection.
- [int_t](#) [tls_bits](#) ([TLS](#) *tls)
Provide the number of secret bits, and thus strength, of the TLS connection cipher suite.
- [chr_t](#) * [tls_version](#) ([TLS](#) *tls)
Provide the SSL/TLS/DTLS version as a string constant.
- [chr_t](#) * [tls_suite](#) ([TLS](#) *tls)
Provide the SSL/TLS/DTLS cipher suite as a string constant.
- int [tls_status](#) ([TLS](#) *tls)
Checks whether a TLS connection has been shut down or not.
- [stringer_t](#) * [tls_error](#) ([TLS](#) *tls, [int_t](#) code, [stringer_t](#) *output)
Consolidate the complicated logic associated with handling SSL_read/SSL_write calls which result in 0, or a negative number.
- int [tls_continue](#) ([TLS](#) *tls, int result, int syserror)
Return -1 if the connection is invalid, 0 if the operation should be retried, or a positive number indicating the number of bytes processed.
- int [tls_read](#) ([TLS](#) *tls, void *buffer, int length, [bool_t](#) block)
Read data from a TLS connection.
- int [tls_write](#) ([TLS](#) *tls, const void *buffer, int length, [bool_t](#) block)
Write data to an open TLS connection.
- int [tls_print](#) ([TLS](#) *tls, const char *format, va_list args)
Write formatted data to an TLS connection.

5.266.1 Function Documentation

5.266.1.1 `int_t tls_bits (TLS * tls)`

Provide the number of secret bits, and thus strength, of the TLS connection cipher suite. [tls.c](#)

See also:

`SSL_CIPHER_get_bits()`

Definition at line 369 of file `tls.c`.

References `SSL_CIPHER_get_bits_d`, and `SSL_get_current_cipher_d`.

5.266.1.2 `stringer_t* tls_cipher (TLS * tls, stringer_t * output)`

Return the name of the TLS cipher suite assoicated with a connection.

See also:

`SSL_CIPHER_get_name()`

Parameters:

tls the TLS connection being inspected.

output a managed string to receive the encoded output; if passed as NULL, an output buffer will be allocated which must be freed by the caller.

Returns:

a pointer to the result, or NULL if an error occurs.

Definition at line 341 of file `tls.c`.

References `NULLER`, `PLACER`, `SSL_CIPHER_get_name_d`, `SSL_get_current_cipher_d`, `SSL_get_version_d`, `st_cmp_cs_eq()`, `st_output()`, and `st_sprint()`.

Referenced by `client_read()`, `client_read_line()`, and `client_write()`.

5.266.1.3 `void* tls_client_alloc (int_t sockd)`

Establish an TLS client wrapper around a socket descriptor.

Parameters:

sockd the file descriptor of the socket to have its transport security level upgraded.

Returns:

NULL on failure or a pointer to the SSL handle of the file descriptor if SSL negotiation was successful.

LOW: Add requisite config options and sandbox resources to verify server TLS certificates.

Definition at line 240 of file `tls.c`.

References `BIO_new_socket_d`, `ERR_clear_error_d`, `log_pedantic`, `MEMORYBUF`, `SSL_connect_d`, `SSL_CTX_ctrl_d`, `SSL_CTX_free_d`, `SSL_CTX_new_d`, `SSL_CTX_set_verify_d`, `ssl_error_string()`, `SSL_free_d`, `SSL_get_error_d`, `SSL_new_d`, `SSL_set_bio_d`, `SSL_set_connect_state_d`, and `SSLv23_client_method_d`.

Referenced by `client_secure()`.

5.266.1.4 int tls_continue (TLS * *tls*, int *result*, int *syserror*)

Return -1 if the connection is invalid, 0 if the operation should be retried, or a positive number indicating the number of bytes processed.

Definition at line 496 of file `tls.c`.

References `ERR_error_string_n_d`, `ERR_get_error_d`, `errno_name()`, `log_pedantic`, `MEMORYBUF`, `SSL_get_error_d`, and `status`.

Referenced by `tls_read()`, and `tls_write()`.

5.266.1.5 stringer_t* tls_error (TLS * *tls*, int_t *code*, stringer_t * *output*)

Consolidate the complicated logic associated with handling `SSL_read/SSL_write` calls which result in 0, or a negative number.

Definition at line 436 of file `tls.c`.

References `ERR_error_string_n_d`, `ERR_get_error_d`, `log_pedantic`, `MEMORYBUF`, `SSL_get_error_d`, `st_output()`, and `st_sprint()`.

Referenced by `client_read()`, `client_read_line()`, and `client_write()`.

5.266.1.6 void tls_free (TLS * *tls*)

Shutdown and free an TLS connection.

Parameters:

TLS the TLS connection to be shut down.

Returns:

This function returns no value.

Definition at line 316 of file `tls.c`.

References `ERR_remove_thread_state_d`, `log_pedantic`, `SSL_free_d`, and `SSL_shutdown_d`.

Referenced by `client_close()`, `con_destroy()`, and `protocol_secure()`.

5.266.1.7 int tls_print (TLS * *tls*, const char * *format*, va_list *args*)

Write formatted data to an TLS connection.

Parameters:

tls the SSL connection to which the data will be written.

format a format string specifying the data to be written to the SSL connection.

va_list a variable argument list containing the data parameters associated with the format string.

Returns:

-1 on error, or the number of bytes written to the SSL connection.

Definition at line 625 of file `tls.c`.

References `bytes`, `length`, `mm_alloc()`, `mm_free()`, `status`, and `tls_write()`.

5.266.1.8 int tls_read (TLS * *tls*, void * *buffer*, int *length*, bool_t *block*)

Read data from a TLS connection.

Parameters:

- tls* the TLS connection from which the data will be read.
- buffer* a pointer to the buffer where the read data will be stored.
- length* the length, in bytes, of the amount of data to be read.
- block* a boolean variable specifying whether the read operation should block.

Returns:

- 1 on failure, 0 if the connection has been closed, or the number of bytes read from the connection on success.

Definition at line 556 of file `tls.c`.

References `ERR_clear_error_d`, `log_pedantic`, `SSL_read_d`, and `tls_continue()`.

Referenced by `client_read()`, `client_read_line()`, `con_read()`, and `con_read_line()`.

5.266.1.9 TLS* `tls_server_alloc` (void * *server*, int *sockd*, int *flags*)

Create a TLS session for a file descriptor, and accept the client TLS/SSL handshake.

See also:

- `SSL_accept()`
- `BIO_new_socket()`

Parameters:

- server* a server object which contains the underlying SSL context.
- sockd* the file descriptor of the TCP connection to be made SSL-ready.
- flags* passed to `BIO_new_socket()`, determines whether the socket is shut down when the BIO is freed.

Definition at line 158 of file `tls.c`.

References `BIO_new_socket_d`, `server_t::context`, `ERR_clear_error_d`, `errno_name()`, `log_pedantic`, `MEMORYBUF`, `SSL_accept_d`, `ssl_error_string()`, `SSL_free_d`, `SSL_get_error_d`, `SSL_new_d`, `SSL_set_accept_state_d`, `SSL_set_bio_d`, `status`, and `server_t::tls`.

Referenced by `imap_starttls()`, `pop_starttls()`, `protocol_secure()`, and `smtp_starttls()`.

5.266.1.10 bool_t `tls_server_create` (void * *server*, uint_t *security_level*)

Setup an TLS CTX for a server.

Note:

- The server is passed as a void pointer because the provider layer doesn't comprehend protocol specific server instances.

Parameters:

- server_t* the TLS context will be assigned to the provided server context.
- security_level* an integer which will be used to control how the TLS context is configured: 0 = accept any type of SSL or TLS protocol version, and offer broad cipher support. 1 = require TLSv1 and above, refuse SSLv2 and SSLv3 connections, use any reasonably secure cipher. (recommened) 2 = require TLSv1 and above, refuse SSLv2 and SSLv3 connections, only use ciphers which provide forward secrecy. 3 = require TLSv1.2 and limit the cipher list to ECDHE-RSA-AES256-GCM-SHA384 or ECDHE-RSA-CHACHA20-POLY1305 as required by the specifications.

Definition at line 24 of file `tls.c`.

References `server_t::certificate`, `server_t::context`, `magma_t::cryptography`, `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_static_2048()`, `dh_static_4096()`, `magma_t::iface`, `log_critical`, `magma`, `MAGMA_CIPHERS_GENERIC`, `MAGMA_CIPHERS_HIGH`, `MAGMA_CIPHERS_MEDIUM`, `SSL_CTX_check_private_key_d`, `SSL_CTX_ctrl_d`, `SSL_CTX_new_d`, `SSL_CTX_set_cipher_list_d`, `SSL_CTX_set_tmp_dh_callback_d`, `SSL_CTX_set_tmp_ecdh_callback_d`, `SSL_CTX_set_verify_d`, `SSL_CTX_use_certificate_chain_file_d`, `SSL_CTX_use_PrivateKey_file_d`, `ssl_ecdh_exchange_callback()`, `SSLv23_server_method_d`, and `server_t::tls`.

Referenced by `servers_encryption_start()`.

5.266.1.11 `void tls_server_destroy (void * server)`

Destroy an TLS context associated with a server.

Parameters:

server the server to be deactivated.

Returns:

This function returns no value.

Definition at line 135 of file `tls.c`.

References `server_t::context`, `log_pedantic`, `SSL_CTX_free_d`, and `server_t::tls`.

Referenced by `servers_encryption_stop()`.

5.266.1.12 `int tls_status (TLS * tls)`

Checks whether a TLS connection has been shut down or not.

See also:

`SSL_get_shutdown()`

Parameters:

tls the TLS connection to be shut down.

Returns:

0 if the connection is alive and well, or `SSL_SENT_SHUTDOWN/SSL_RECEIVED_SHUTDOWN`

Definition at line 421 of file `tls.c`.

References `SSL_get_shutdown_d`.

Referenced by `client_status()`, and `con_status()`.

5.266.1.13 `chr_t* tls_suite (TLS * tls)`

Provide the SSL/TLS/DTLS cipher suite as a string constant.

See also:

`SSL_CIPHER_get_name()`

Note:

The RFC version of the TLS cipher suite is available through the `SSL_CIPHER_standard_name()` function.

Definition at line 403 of file `tls.c`.

References `SSL_CIPHER_get_name_d`, and `SSL_get_current_cipher_d`.

5.266.1.14 `chr_t* tls_version (TLS * tls)`

Provide the SSL/TLS/DTLS version as a string constant.

See also:

`SSL_get_version()`

Definition at line 385 of file `tls.c`.

References `NULLER`, `PLACER`, `SSL_get_current_cipher_d`, `SSL_get_version_d`, and `st_cmp_cs_eq()`.

5.266.1.15 `int tls_write (TLS * tls, const void * buffer, int length, bool_t block)`

Write data to an open TLS connection.

Parameters:

tls the TLS connection to which the data will be written.

buffer a pointer to the buffer containing the data to be written.

length the length, in bytes, of the data to be written.

Returns:

-1 on error, or the number of bytes written to the TLS connection.

Definition at line 592 of file `tls.c`.

References `ERR_clear_error_d`, `log_pedantic`, `SSL_write_d`, and `tls_continue()`.

Referenced by `client_write()`, `con_write_bl()`, and `tls_print()`.

5.267 src/providers/database/database.h File Reference

Defines

- #define [ISNULL](#)(b) (my_bool *)&((my_bool){ b })

Typedefs

- typedef char [table_t](#)
- typedef char [row_t](#)

Functions

- [bool_t lib_load_mysql](#) (void)
mysql.c
- const char * [lib_version_mysql](#) ()
Return the version string of libmysql.
- const char * [serv_charset_mysql](#) (void)
- const char * [serv_schema_mysql](#) (void)
- const char * [serv_type_mysql](#) (void)
Determine whether the mysql library in use is embedded or not.
- const char * [serv_version_mysql](#) (void)
Return the server version string of the mysql database.
- [uint_t sql_errno](#) (MYSQL *mysql)
Get the last error number for a mysql connection.
- const [chr_t](#) * [sql_error](#) (MYSQL *mysql)
Get a human-readable error message for a mysql connection.
- MYSQL * [sql_open](#) ([bool_t](#) silent)
Open up a new mysql connection to the configured database server.
- [int_t sql_ping](#) ([uint32_t](#) connection)
Ping the MySQL server via the provided connection. If the underlying TCP/IP socket has timed out, the ping function will will cause the MySQL client library to reestablish the connection. Because a new connection is created the collection of prepared statements will need to be reinitialized.
- [bool_t sql_start](#) (void)
Load up the mysql subsystem.
- void [sql_stop](#) (void)
Shutdown and free the pool of mysql connections.
- [bool_t sql_thread_start](#) (void)
Initialize mysql thread specific variables for the calling thread.
- void [sql_thread_stop](#) (void)
Prepare the thread for exiting to destroy mysql thread specific variables.

- `int64_t sql_insert (stringer_t *query)`
query.c
- `int64_t sql_insert_conn (stringer_t *query, uint32_t connection)`
- `int64_t sql_num_rows (stringer_t *query)`
- `int64_t sql_num_rows_conn (stringer_t *query, uint32_t connection)`
- `int64_t sql_query (stringer_t *query)`
Pull a connection from the sql pool and execute a mysql statement.
- `int64_t sql_query_conn (stringer_t *query, uint32_t connection)`
Execute a mysql statement.
- `MYSQL_RES * sql_query_res (stringer_t *query)`
- `MYSQL_RES * sql_query_res_conn (stringer_t *query, uint32_t connection)`
- `int64_t sql_write (stringer_t *query)`
- `int64_t sql_write_conn (stringer_t *query, uint32_t connection)`
- `uint64_t res_bind_create (MYSQL_STMT *stmt, MYSQL_BIND **result)`
results.c
- `void res_bind_free (MYSQL_STMT *stmt, MYSQL_BIND *binding, uint64_t number)`
Free a prepared MySQL statement result set binding.
- `void * res_field_block (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as generic pointer.
- `bool_t res_field_bool (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as a boolean.
- `uint64_t res_field_count (table_t *table)`
Return the number of fields stored in the database results table.
- `double_t res_field_double (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as a double.
- `float_t res_field_float (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as a float.
- `row_t * res_field_generic (row_t *row, uint64_t field, size_t typesize)`
- `int16_t res_field_int16 (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as a signed 16-bit integer.
- `int32_t res_field_int32 (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as a signed 32-bit integer.
- `int64_t res_field_int64 (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as a signed 64-bit integer.
- `int8_t res_field_int8 (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as a signed 8-bit integer.
- `size_t res_field_length (row_t *row, uint64_t field)`
Get the length of a specified field in a database result row.

- `stringer_t * res_field_string (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as managed string.
- `uint16_t res_field_uint16 (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as an unsigned 16-bit integer.
- `uint32_t res_field_uint32 (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as an unsigned 32-bit integer.
- `uint64_t res_field_uint64 (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as an unsigned 64-bit integer.
- `uint8_t res_field_uint8 (row_t *row, uint64_t field)`
Get the value of a specified field in a database result row as an unsigned 8-bit integer.
- `uint64_t res_row_count (table_t *table)`
Return the number of rows in the MySQL results table.
- `row_t * res_row_get (table_t *table, uint64_t row)`
Retrieve a row from a database results table by index.
- `row_t * res_row_next (table_t *table)`
Return the next row in the database results table and advance the cursor.
- `void res_row_set (row_t *row, chr_t *buffer)`
Set the buffer location for a database result row.
- `bool_t res_row_store (uint64_t num, table_t *table, MYSQL_BIND *binding)`
- `table_t * res_stmt_store (MYSQL_STMT *stmt)`
- `table_t * res_table_alloc (uint64_t rows, uint64_t fields)`
- `void res_table_free (table_t *table)`
Free a MySQL results table.
- `bool_t stmt_bind_param (MYSQL_STMT *group, MYSQL_BIND *bind)`
stmts.c
- `void stmt_close (MYSQL_STMT *local)`
- `uint_t stmt_errno (MYSQL_STMT *local)`
Get the error number associated with a mysql statement.
- `const chr_t * stmt_error (MYSQL_STMT *local)`
- `bool_t stmt_exec (MYSQL_STMT **group, MYSQL_BIND *parameters)`
Execute a prepared mysql statement.
- `int64_t stmt_exec_affected (MYSQL_STMT **group, MYSQL_BIND *parameters)`
Execute a statement on a specified mysql connection and return the number of affected rows.
- `int64_t stmt_exec_affected_conn (MYSQL_STMT **group, MYSQL_BIND *parameters, uint32_t connection)`
Execute a statement on a specified mysql connection and return the number of affected rows.
- `bool_t stmt_exec_conn (MYSQL_STMT **group, MYSQL_BIND *parameters, uint32_t connection)`
Execute a prepared mysql statement over a specified connection.

- [table_t * stmt_get_result](#) (MYSQL_STMT **group, MYSQL_BIND *parameters)
Execute a prepared mysql statement and return the result.
- [table_t * stmt_get_result_conn](#) (MYSQL_STMT **group, MYSQL_BIND *parameters, uint32_t connection)
Execute a prepared mysql statement on a specified connection and return the result.
- [uint64_t stmt_insert](#) (MYSQL_STMT **group, MYSQL_BIND *parameters)
Execute a mysql prepared INSERT or UPDATE statement.
- [uint64_t stmt_insert_conn](#) (MYSQL_STMT **group, MYSQL_BIND *parameters, uint32_t connection)
Execute a mysql prepared INSERT or UPDATE statement on a specified connection.
- [MYSQL_STMT * stmt_open](#) (MYSQL *mysql)
Initialize a mysql prepared statement.
- [bool_t stmt_prepare](#) (MYSQL_STMT *group, const char *query, unsigned long [length](#))
- [bool_t stmt_rebuild](#) (uint32_t connection)
- [MYSQL_STMT * stmt_reset](#) (MYSQL_STMT **group, uint32_t connection)
- [bool_t stmt_start](#) (void)
Initialize the global array of mysql prepared statements.
- [void stmt_stop](#) (void)
- [int64_t tran_commit](#) (int64_t transaction)
transaction.c
- [int64_t tran_rollback](#) (int64_t transaction)
Rollback a pending transaction in the mysql database.
- [int64_t tran_start](#) (void)
Pull a connection from the SQL pool and start a transaction.

Variables

- [pool_t * sql_pool](#)

5.267.1 Define Documentation

5.267.1.1 #define ISNULL(b) (my_bool *)&((my_bool){ b })

Definition at line 63 of file database.h.

Referenced by [contact_update\(\)](#), [mail_db_insert_duplicate_message\(\)](#), and [mail_db_insert_message\(\)](#).

5.267.2 Typedef Documentation

5.267.2.1 typedef char row_t

Definition at line 60 of file database.h.

5.267.2.2 typedef char table_t

Definition at line 59 of file database.h.

5.267.3 Function Documentation

5.267.3.1 `bool_t lib_load_mysql (void)`

[mysql.c](#) [mysql.c](#)

Returns:

true on success or false on failure.

Definition at line 494 of file `mysql.c`.

References `lib_symbols()`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.267.3.2 `const char* lib_version_mysql (void)`

Return the version string of libmysql.

Returns:

a pointer to a character string containing the libmysql version information.

Definition at line 471 of file `mysql.c`.

References `mysql_get_client_version_d`, and `sql`.

Referenced by `lib_load()`.

5.267.3.3 `uint64_t res_bind_create (MYSQL_STMT * stmt, MYSQL_BIND ** result)`

[results.c](#)

Definition at line 348 of file `results.c`.

References `length`, `log_info`, `mm_alloc()`, `mysql_fetch_field_d`, `mysql_free_result_d`, `mysql_num_fields_d`, `mysql_stmt_error_d`, `mysql_stmt_result_metadata_d`, and `res_bind_free()`.

Referenced by `res_stmt_store()`.

5.267.3.4 `void res_bind_free (MYSQL_STMT * stmt, MYSQL_BIND * binding, uint64_t number)`

Free a prepared MySQL statement result set binding.

Parameters:

stmt the input prepared MySQL statement.

binding the binding for the result set.

number the number of fields in the result set.

Returns:

This function does not return a value.

Definition at line 290 of file `results.c`.

References `length`, `mm_cleanup`, `mm_free()`, and `mysql_stmt_bind_result_d`.

Referenced by `res_bind_create()`, and `res_stmt_store()`.

5.267.3.5 void* res_field_block (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as generic pointer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a generic pointer, or NULL on failure.

Definition at line 43 of file results.c.

References res_field_generic().

Referenced by auth_data_fetch(), config_load_database_settings(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_fetch_alerts(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), meta_data_fetch_shard(), meta_data_fetch_user(), smtp_check_receive_quota(), smtp_fetch_inbound(), user_config_fetch(), and warehouse_fetch_domains().

5.267.3.6 bool_t res_field_bool (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as a boolean.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a boolean, or false on failure.

Definition at line 216 of file results.c.

References res_field_generic().

5.267.3.7 uint64_t res_field_count (table_t * table)

Return the number of fields stored in the database results table.

Parameters:

table the input database results table.

Returns:

0 on failure, or the number of table fields on success.

Definition at line 415 of file results.c.

References log_info.

Referenced by res_row_store().

5.267.3.8 `double_t res_field_double (row_t * row, uint64_t field)`

Get the value of a specified field in a database result row as a double.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a double, or 0 on failure.

Definition at line 86 of file results.c.

References `res_field_generic()`.

5.267.3.9 `float_t res_field_float (row_t * row, uint64_t field)`

Get the value of a specified field in a database result row as a float.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a float, or 0 on failure.

Definition at line 99 of file results.c.

References `res_field_generic()`.

5.267.3.10 `row_t* res_field_generic (row_t * row, uint64_t field, size_t typesize)`

Definition at line 10 of file results.c.

References `log_info`.

Referenced by `res_field_block()`, `res_field_bool()`, `res_field_double()`, `res_field_float()`, `res_field_int16()`, `res_field_int32()`, `res_field_int64()`, `res_field_int8()`, `res_field_uint16()`, `res_field_uint32()`, `res_field_uint64()`, and `res_field_uint8()`.

5.267.3.11 `int16_t res_field_int16 (row_t * row, uint64_t field)`

Get the value of a specified field in a database result row as a signed 16-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a signed 16-bit integer, or 0 on failure.

Definition at line 190 of file results.c.

References `res_field_generic()`.

5.267.3.12 int32_t res_field_int32 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as a signed 32-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a signed 32-bit integer, or 0 on failure.

Definition at line 177 of file results.c.

References res_field_generic().

5.267.3.13 int64_t res_field_int64 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as a signed 64-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a signed 64-bit integer, or 0 on failure.

Definition at line 164 of file results.c.

References res_field_generic().

5.267.3.14 int8_t res_field_int8 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as a signed 8-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a signed 8-bit integer, or 0 on failure.

Definition at line 203 of file results.c.

References res_field_generic().

Referenced by auth_data_fetch(), meta_data_fetch_user(), smtp_fetch_authorization(), smtp_fetch_inbound(), teacher_data_fetch(), and warehouse_fetch_domains().

5.267.3.15 size_t res_field_length (row_t * row, uint64_t field)

Get the length of a specified field in a database result row.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the length of the specified result row, or 0 on failure.

Definition at line 229 of file results.c.

References log_info.

Referenced by auth_data_fetch(), config_load_database_settings(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_fetch_alerts(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), meta_data_fetch_shard(), meta_data_fetch_user(), smtp_check_receive_quota(), smtp_fetch_inbound(), user_config_fetch(), and warehouse_fetch_domains().

5.267.3.16 stringer_t* res_field_string (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as managed string.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a managed string, or NULL on failure.

Definition at line 56 of file results.c.

References length, log_info, and st_import().

Referenced by auth_data_fetch(), meta_data_fetch_all_tags(), meta_data_fetch_message_tags(), meta_data_fetch_user(), register_data_fetch_blocklist(), smtp_fetch_authorization(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_rollout(), teacher_data_fetch(), and warehouse_fetch_patterns().

5.267.3.17 uint16_t res_field_uint16 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as an unsigned 16-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as an unsigned 16-bit integer, or 0 on failure.

Definition at line 138 of file results.c.

References res_field_generic().

5.267.3.18 uint32_t res_field_uint32 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as an unsigned 32-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as an unsigned 32-bit integer, or 0 on failure.

Definition at line 125 of file results.c.

References `res_field_generic()`.

Referenced by `auth_data_fetch()`, `magma_folder_fetch()`, `meta_data_fetch_folder_messages()`, `meta_data_fetch_folders()`, `meta_data_fetch_messages()`, `smtp_fetch_authorization()`, `smtp_fetch_inbound()`, and `smtp_rollout()`.

5.267.3.19 uint64_t res_field_uint64 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as an unsigned 64-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as an unsigned 64-bit integer, or 0 on failure.

Definition at line 112 of file results.c.

References `res_field_generic()`.

Referenced by `auth_data_fetch()`, `config_fetch_host_number()`, `contact_details_fetch()`, `contacts_fetch()`, `magma_folder_fetch()`, `meta_data_acknowledge_alert()`, `meta_data_fetch_alerts()`, `meta_data_fetch_folder_messages()`, `meta_data_fetch_folders()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_messages()`, `smtp_check_receive_quota()`, `smtp_check_transmit_quota()`, `smtp_fetch_authorization()`, `smtp_fetch_inbound()`, `smtp_rollout()`, `statistics_refresh()`, `teacher_data_fetch()`, and `user_config_fetch()`.

5.267.3.20 uint8_t res_field_uint8 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as an unsigned 8-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as an unsigned 8-bit integer, or 0 on failure.

Definition at line 151 of file results.c.

References `res_field_generic()`.

Referenced by `meta_data_fetch_mailbox_aliases()`, and `meta_data_fetch_shard()`.

5.267.3.21 `uint64_t res_row_count (table_t * table)`

Return the number of rows in the MySQL results table.

Note:

The row count is provided for a one-indexed table.

Parameters:

table the input MySQL results table.

Returns:

0 on error, or the row count on success.

Definition at line 432 of file results.c.

References log_info.

Referenced by auth_data_fetch(), config_load_database_settings(), meta_data_fetch_message_tags(), res_row_store(), smtp_check_authorized_from(), and smtp_fetch_inbound().

5.267.3.22 `row_t* res_row_get (table_t * table, uint64_t row)`

Retrieve a row from a database results table by index.

Note:

This function operates on a 0-indexed table.

Parameters:

table the input database results table. row the row # to be fetched.

Returns:

NULL on failure, or a pointer to a result row on success.

Definition at line 469 of file results.c.

References log_info.

Referenced by config_load_database_settings(), res_row_next(), and res_row_store().

5.267.3.23 `row_t* res_row_next (table_t * table)`

Return the next row in the database results table and advance the cursor.

Parameters:

table the input mysql results table.

Returns:

a pointer to the next result row in the table.

Definition at line 490 of file results.c.

References count, and res_row_get().

Referenced by auth_data_fetch(), config_fetch_host_number(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_acknowledge_alert(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(),

meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_message_tags(), meta_data_fetch_messages(), meta_data_fetch_shard(), meta_data_fetch_user(), register_data_check_username(), register_data_fetch_blocklist(), smtp_check_receive_quota(), smtp_check_transmit_quota(), smtp_fetch_authorization(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_rollout(), statistics_refresh(), teacher_data_fetch(), user_config_fetch(), warehouse_fetch_domains(), and warehouse_fetch_patterns().

5.267.3.24 void res_row_set (row_t * row, chr_t * buffer)

Set the buffer location for a database result row.

Parameters:

- row* the input database result row.
- buffer* the buffer to be assigned to the target row.

Returns:

This function returns no value.

Definition at line 449 of file results.c.

References log_info.

Referenced by res_row_store().

5.267.3.25 bool_t res_row_store (uint64_t num, table_t * table, MYSQL_BIND * binding)

Definition at line 510 of file results.c.

References length, log_info, mm_alloc(), mm_copy(), res_field_count(), res_row_count(), res_row_get(), and res_row_set().

Referenced by res_stmt_store().

5.267.3.26 table_t* res_stmt_store (MYSQL_STMT * stmt)

Definition at line 579 of file results.c.

References log_info, mysql_stmt_bind_result_d, mysql_stmt_error_d, mysql_stmt_fetch_d, mysql_stmt_free_result_d, mysql_stmt_num_rows_d, mysql_stmt_store_result_d, res_bind_create(), res_bind_free(), res_row_store(), res_table_alloc(), and res_table_free().

Referenced by stmt_get_result_conn().

5.267.3.27 table_t* res_table_alloc (uint64_t rows, uint64_t fields)

Definition at line 316 of file results.c.

References log_info, and mm_alloc().

Referenced by res_stmt_store().

5.267.3.28 void res_table_free (table_t * table)

Free a MySQL results table.

Parameters:

- table* the MySQL results table to be freed.

Returns:

This function does not return a value.

Definition at line 250 of file results.c.

References `log_info`, and `mm_free()`.

Referenced by `auth_data_fetch()`, `config_fetch_host_number()`, `config_load_database_settings()`, `contact_details_fetch()`, `contacts_fetch()`, `magma_folder_fetch()`, `meta_data_acknowledge_alert()`, `meta_data_fetch_alerts()`, `meta_data_fetch_all_tags()`, `meta_data_fetch_folder_messages()`, `meta_data_fetch_folders()`, `meta_data_fetch_keys()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_message_tags()`, `meta_data_fetch_messages()`, `meta_data_fetch_shard()`, `meta_data_fetch_user()`, `register_data_check_username()`, `register_data_fetch_blocklist()`, `res_stmt_store()`, `smtp_check_authorized_from()`, `smtp_check_receive_quota()`, `smtp_check_transmit_quota()`, `smtp_fetch_authorization()`, `smtp_fetch_autoreply()`, `smtp_fetch_inbound()`, `smtp_rollout()`, `statistics_refresh()`, `teacher_data_fetch()`, `user_config_fetch()`, `warehouse_fetch_domains()`, and `warehouse_fetch_patterns()`.

5.267.3.29 `const char* serv_charset_mysql (void)`

Definition at line 415 of file mysql.c.

References `mm_wipe()`, `mysql_character_set_name_d`, `mysql_close_d`, `sql`, and `sql_open()`.

Referenced by `lib_load()`.

5.267.3.30 `const char* serv_schema_mysql (void)`

Definition at line 431 of file mysql.c.

References `mm_wipe()`, `mysql_close_d`, `sql`, and `sql_open()`.

Referenced by `lib_load()`.

5.267.3.31 `const char* serv_type_mysql (void)`

Determine whether the mysql library in use is embedded or not.

Returns:

`sql.type_embed` ("Embedded") or `sql.type_serv` ("MySQL");

Definition at line 401 of file mysql.c.

References `mysql_embedded_d`, and `sql`.

Referenced by `lib_load()`.

5.267.3.32 `const char* serv_version_mysql (void)`

Return the server version string of the mysql database.

Returns:

a pointer to a character string containing the mysql server version information.

Definition at line 451 of file mysql.c.

References `mm_wipe()`, `mysql_close_d`, `mysql_get_server_info_d`, `sql`, and `sql_open()`.

Referenced by `lib_load()`.

5.267.3.33 `uint_t sql_errno (MYSQL *mysql)`

Get the last error number for a mysql connection.

Parameters:

mysql a pointer to the MYSQL object of the connection to be queried.

Returns:

0 on failure, or the last mysql error message for the connection on success.

Definition at line 163 of file mysql.c.

References mysql_errno_d.

Referenced by stmt_close().

5.267.3.34 const chr_t* sql_error (MYSQL * *mysql*)

Get a human-readable error message for a mysql connection.

Parameters:

mysql a pointer to the MYSQL object of the connection to be queried.

Returns:

NULL on failure, or a pointer to a null-terminated string with the error message on success.

Definition at line 177 of file mysql.c.

References mysql_errno_d, and mysql_error_d.

Referenced by sql_open(), sql_ping(), sql_query_conn(), stmt_close(), stmt_open(), tran_commit(), tran_rollback(), and tran_start().

5.267.3.35 int64_t sql_insert (stringer_t * *query*)

[query.c](#)

Definition at line 124 of file query.c.

References log_pedantic, PL_RESERVED, pool_pull(), pool_release(), sql_insert_conn(), and sql_pool.

5.267.3.36 int64_t sql_insert_conn (stringer_t * *query*, uint32_t *connection*)

Definition at line 48 of file query.c.

References log_pedantic, mysql_error_d, mysql_insert_id_d, mysql_real_query_d, pool_get_obj(), sql_pool, st_data_get(), and st_length_get().

Referenced by sql_insert().

5.267.3.37 int64_t sql_num_rows (stringer_t * *query*)

Definition at line 109 of file query.c.

References log_pedantic, PL_RESERVED, pool_pull(), pool_release(), sql_num_rows_conn(), and sql_pool.

5.267.3.38 int64_t sql_num_rows_conn (stringer_t * *query*, uint32_t *connection*)

Definition at line 27 of file query.c.

References log_pedantic, mysql_error_d, mysql_free_result_d, mysql_num_rows_d, mysql_real_query_d, mysql_store_result_d, pool_get_obj(), sql_pool, st_data_get(), and st_length_get().

Referenced by sql_num_rows().

5.267.3.39 MySQL* `sql_open (bool_t silent)`

Open up a new mysql connection to the configured database server.

Note:

The reconnect option will automatically be set on all new mysql connections.

Parameters:

silent if true, suppress logging of failure messages for this function.

Returns:

NULL on failure, or a pointer to a MYSQL objection for the newly established connection on success.

Definition at line 216 of file mysql.c.

References magma_t::database, magma_t::iface, log_critical, magma, mysql_close_d, mysql_init_d, mysql_options_d, mysql_real_connect_d, and sql_error().

Referenced by serv_charset_mysql(), serv_schema_mysql(), serv_version_mysql(), and sql_start().

5.267.3.40 `int_t sql_ping (uint32_t connection)`

Ping the MySQL server via the provided connection. If the underlying TCP/IP socket has timed out, the ping function will will cause the MySQL client library to reestablish the connection. Because a new connection is created the collection of prepared statements will need to be reinitialized.

Warning:

If the connection timed out, the ping will trigger a reconnect, which invalidates the prepared statement references.

```
{.c}
if (sql_ping(connection) < 0 || !stmt_rebuild(connection)) { log_error("Invalid
    database connection."); return; }
```

Parameters:

connection The specific connection inside the pool that should be used for the ping.

Returns:

Returns 1 if the library performed an automatic reconnect, 0 if the connection is active, and -1 if the reconnect failed.

Definition at line 296 of file mysql.c.

References log_error, mysql_ping_d, mysql_thread_id_d, pool_get_obj(), sql_error(), and sql_pool.

Referenced by dspam_check(), dspam_train(), and stmt_reset().

5.267.3.41 `int64_t sql_query (stringer_t * query)`

Pull a connection from the sql pool and execute a mysql statement.

See also:

[sql_query_conn\(\)](#)

Parameters:

query the mysql statement to be executed.

Returns:

0 on success, or non-zero on failure.

Definition at line 160 of file query.c.

References log_info, PL_RESERVED, pool_pull(), pool_release(), sql_pool, and sql_query_conn().

5.267.3.42 int64_t sql_query_conn (stringer_t * *query*, uint32_t *connection*)

Execute a mysql statement.

See also:

mysql_real_query()

Parameters:

query the mysql statement to be executed.

connection a connection id for the underlying mysql session.

Returns:

0 on success, or a non-zero number on error.

Definition at line 83 of file query.c.

References log_pedantic, mysql_real_query_d, pool_get_obj(), sql_error(), sql_pool, st_char_get(), st_data_get(), st_length_get(), and st_length_int().

Referenced by sql_query(), tran_commit(), tran_rollback(), and tran_start().

5.267.3.43 MYSQL_RES* sql_query_res (stringer_t * *query*)

Definition at line 94 of file query.c.

References log_pedantic, PL_RESERVED, pool_pull(), pool_release(), sql_pool, and sql_query_res_conn().

5.267.3.44 MYSQL_RES* sql_query_res_conn (stringer_t * *query*, uint32_t *connection*)

Definition at line 10 of file query.c.

References log_pedantic, mysql_error_d, mysql_real_query_d, mysql_store_result_d, pool_get_obj(), sql_pool, st_data_get(), and st_length_get().

Referenced by sql_query_res().

5.267.3.45 bool_t sql_start (void)

Load up the mysql subsystem.

Note:

This function will check that all necessary database parameters have been supplied, and that the supplied mysql library version is thread safe and initialized. It will also initialize the pool of database connections, and

Returns:

true on success or false on failure.

Definition at line 324 of file mysql.c.

References magma_t::database, magma_t::iface, log_critical, magma, mysql_server_init_d, mysql_thread_safe_d, ns_empty(), pool_alloc(), pool_set_obj(), sql_open(), sql_pool, sql_stop(), and stmt_start().

Referenced by process_start().

5.267.3.46 void sql_stop (void)

Shutdown and free the pool of mysql connections.

Returns:

This function returns no value.

Definition at line 189 of file mysql.c.

References magma_t::database, magma_t::iface, magma, my_once_free_d, mysql_close_d, mysql_server_end_d, pool_free(), pool_get_obj(), sql_pool, and stmt_stop().

Referenced by process_stop(), and sql_start().

5.267.3.47 bool_t sql_thread_start (void)

Initialize mysql thread specific variables for the calling thread.

Note:

mysql_thread_init()

Returns:

0 on success or non-zero on error.

Definition at line 388 of file mysql.c.

References log_error, and mysql_thread_init_d.

Referenced by thread_start().

5.267.3.48 void sql_thread_stop (void)

Prepare the thread for exiting to destroy mysql thread specific variables.

Note:

mysql_thread_end()

Returns:

This function returns no value.

Definition at line 378 of file mysql.c.

References mysql_thread_end_d.

Referenced by thread_stop().

5.267.3.49 int64_t sql_write (stringer_t * query)

Definition at line 139 of file query.c.

References log_pedantic, PL_RESERVED, pool_pull(), pool_release(), sql_pool, and sql_write_conn().

5.267.3.50 int64_t sql_write_conn (stringer_t * *query*, uint32_t *connection*)

Definition at line 62 of file query.c.

References log_pedantic, mysql_affected_rows_d, mysql_error_d, mysql_real_query_d, pool_get_obj(), sql_pool, st_data_get(), and st_length_get().

Referenced by sql_write().

5.267.3.51 bool_t stmt_bind_param (MYSQL_STMT * *group*, MYSQL_BIND * *bind*)

[stmts.c](#)

Definition at line 251 of file stmts.c.

References log_critical, log_pedantic, mysql_stmt_bind_param_d, and stmt_error().

Referenced by stmt_exec_affected_conn(), stmt_exec_conn(), stmt_get_result_conn(), and stmt_insert_conn().

5.267.3.52 void stmt_close (MYSQL_STMT * *local*)

Definition at line 68 of file stmts.c.

References log_critical, mysql_stmt_close_d, sql_errno(), and sql_error().

Referenced by stmt_rebuild(), and stmt_stop().

5.267.3.53 uint_t stmt_errno (MYSQL_STMT * *local*)

Get the error number associated with a mysql statement.

Parameters:

local a pointer to the input mysql statement object.

Returns:

the errno associated with the specified mysql statement.

Definition at line 17 of file stmts.c.

References mysql_errno_d, and mysql_stmt_errno_d.

5.267.3.54 const chr_t* stmt_error (MYSQL_STMT * *local*)

Return the error string for a mysql statement.

Parameters:

local A single MYSQL_STMT* parameter.

Returns:

This function returns the mysql error string, or NULL if unable to retrieve it.

Definition at line 34 of file stmts.c.

References mysql_errno_d, mysql_error_d, mysql_stmt_errno_d, and mysql_stmt_error_d.

Referenced by stmt_bind_param(), stmt_exec_affected_conn(), stmt_exec_conn(), stmt_get_result_conn(), stmt_insert_conn(), and stmt_prepare().

5.267.3.55 bool_t stmt_exec (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*)

Execute a prepared mysql statement.

See also:

[stmt_exec_conn\(\)](#)

Parameters:

group the mysql prepared statement to be executed.

parameters the parameters to be bound to the statement.

Returns:

false on failure, or true on success.

Definition at line 307 of file stmts.c.

References [log_info](#), [PL_RESERVED](#), [pool_pull\(\)](#), [pool_release\(\)](#), [sql_pool](#), and [stmt_exec_conn\(\)](#).

Referenced by [mail_db_hide_message\(\)](#), [meta_data_flags_add\(\)](#), [meta_data_flags_remove\(\)](#), [meta_data_flags_replace\(\)](#), [smtp_update_receive_stats\(\)](#), [smtp_update_transmission_stats\(\)](#), and [teacher_data_delete\(\)](#).

5.267.3.56 int64_t stmt_exec_affected (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*)

Execute a statement on a specified mysql connection and return the number of affected rows.

Note:

Technically the number of affected rows is a my_ulonglong value, but since we retrun -1 when an error occurs, we are using the signed equivalent, int64_t. This means if more than INT64_MAX (or 9,223,372,036,854,775,807) rows are ever altered, the function will return INT64_MAX (or 9,223,372,036,854,775,807) instead.

See also:

[mysql_stmt_affected_rows\(\)](#)

Parameters:

group the prepared mysql statement to be executed.

parameters the parameters to be bound to the prepared statement.

Returns:

-1 on failure, 0 if executed successfully, but no rows are affected, and a positive number with the number of altered rows upon success.

Definition at line 492 of file stmts.c.

References [log_info](#), [PL_RESERVED](#), [pool_pull\(\)](#), [pool_release\(\)](#), [sql_pool](#), and [stmt_exec_affected_conn\(\)](#).

Referenced by [api_endpoint_delete_user\(\)](#), [auth_data_update_legacy\(\)](#), [auth_data_update_lock\(\)](#), [contact_delete\(\)](#), [contact_detail_delete\(\)](#), [contact_detail_upsert\(\)](#), [contact_update\(\)](#), [contact_update_stamp\(\)](#), [magma_folder_delete\(\)](#), [magma_folder_rename\(\)](#), [meta_data_delete_folder\(\)](#), [meta_data_delete_tag\(\)](#), [meta_data_insert_tag\(\)](#), [meta_data_truncate_tags\(\)](#), [meta_data_update_folder_name\(\)](#), [meta_data_update_log\(\)](#), [user_config_delete\(\)](#), and [user_config_upsert\(\)](#).

5.267.3.57 int64_t stmt_exec_affected_conn (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*, uint32_t *connection*)

Execute a statement on a specified mysql connection and return the number of affected rows.

Note:

Technically the number of affected rows is a `my_ulonglong` value, but since we return -1 when an error occurs, we are using the signed equivalent, `int64_t`. This means if more than `INT64_MAX` (or 9,223,372,036,854,775,807) rows are ever altered, the function will return `INT64_MAX` (or 9,223,372,036,854,775,807) instead.

See also:

`mysql_stmt_affected_rows()`

Parameters:

group the prepared mysql statement to be executed.
parameters the parameters to be bound to the prepared statement.
connection the mysql connection id over which the specified statement will be executed.

Returns:

-1 on failure, 0 if executed successfully, but no rows are affected, and a positive number with the number of altered rows upon success.

Definition at line 440 of file `stmts.c`.

References `log_info`, `mysql_stmt_affected_rows_d`, `mysql_stmt_execute_d`, `stmt_bind_param()`, `stmt_error()`, `stmt_reset()`, and `uint64_clamp()`.

Referenced by `mail_db_delete_message()`, `mail_db_update_message_folder()`, `meta_data_acknowledge_alert()`, `meta_data_insert_keys()`, `meta_data_insert_shard()`, `stmt_exec_affected()`, and `tank_delete_object()`.

5.267.3.58 `bool_t stmt_exec_conn (MYSQL_STMT ** group, MYSQL_BIND * parameters, uint32_t connection)`

Execute a prepared mysql statement over a specified connection.

Note:

This function will also reset the prepared statement and bind its parameters.

Parameters:

group the mysql prepared statement to be executed.
parameters the parameters to be bound to the statement.
connection the mysql connection id.

Returns:

false on failure, or true on success.

Definition at line 278 of file `stmts.c`.

References `log_info`, `mysql_stmt_execute_d`, `stmt_bind_param()`, `stmt_error()`, and `stmt_reset()`.

Referenced by `mail_db_insert_duplicate_message()`, `mail_db_insert_message()`, `register_data_insert_user()`, and `stmt_exec()`.

5.267.3.59 `table_t* stmt_get_result (MYSQL_STMT ** group, MYSQL_BIND * parameters)`

Execute a prepared mysql statement and return the result.

Parameters:

group the prepared mysql statement to be executed.
parameters the parameters to be passed with the query.

Returns:

the result of the query, or NULL on failure.

Definition at line 357 of file stmts.c.

References log_info, PL_RESERVED, pool_pull(), pool_release(), sql_pool, and stmt_get_result_conn().

Referenced by auth_data_fetch(), config_fetch_host_number(), config_fetch_settings(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_message_tags(), meta_data_fetch_messages(), meta_data_fetch_user(), register_data_check_username(), register_data_fetch_blocklist(), smtp_check_authorized_from(), smtp_check_receive_quota(), smtp_check_transmit_quota(), smtp_fetch_authorization(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_fetch_rollmessages(), statistics_refresh(), teacher_data_fetch(), user_config_fetch(), warehouse_fetch_domains(), and warehouse_fetch_patterns().

5.267.3.60 table_t* stmt_get_result_conn (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*, uint32_t *connection*)

Execute a prepared mysql statement on a specified connection and return the result.

Parameters:

group the prepared mysql statement to be executed.

parameters the parameters to be passed with the query.

connection the mysql connection identifier.

Returns:

the result of the query, or NULL on failure.

Definition at line 329 of file stmts.c.

References log_info, mysql_stmt_execute_d, res_stmt_store(), stmt_bind_param(), stmt_error(), and stmt_reset().

Referenced by meta_data_acknowledge_alert(), meta_data_fetch_keys(), meta_data_fetch_shard(), and stmt_get_result().

5.267.3.61 uint64_t stmt_insert (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*)

Execute a mysql prepared INSERT or UPDATE statement.

See also:

mysql_stmt_insert_id()

Parameters:

group the mysql prepared statement to be executed.

parameters the parameters to be bound to the prepared statement.

Returns:

0 on failure, or the last LAST_INSERT_ID() value returned for the mysql auto-increment column.

Definition at line 409 of file stmts.c.

References log_info, PL_RESERVED, pool_pull(), pool_release(), sql_pool, and stmt_insert_conn().

Referenced by contact_insert(), magma_folder_insert(), meta_data_insert_folder(), and smtp_insert_spamsig().

5.267.3.62 uint64_t stmt_insert_conn (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*, uint32_t *connection*)

Execute a mysql prepared INSERT or UPDATE statement on a specified connection.

See also:

`mysql_stmt_insert_id()`

Parameters:

group the mysql prepared statement to be executed.

parameters the parameters to be bound to the prepared statement.

connection the underlying connection on which the statement will be executed.

Returns:

0 on failure, or the last LAST_INSERT_ID() value returned for the mysql auto-increment column.

Definition at line 380 of file `stmts.c`.

References `log_info`, `mysql_stmt_execute_d`, `mysql_stmt_insert_id_d`, `stmt_bind_param()`, `stmt_error()`, and `stmt_reset()`.

Referenced by `mail_db_insert_duplicate_message()`, `mail_db_insert_message()`, `register_data_insert_user()`, `stmt_insert()`, and `tank_insert_object()`.

5.267.3.63 MYSQL_STMT* stmt_open (MYSQL * *mysql*)

Initialize a mysql prepared statement.

See also:

`mysql_stmt_init()`

Parameters:

mysql the underlying mysql connection.

Returns:

This function is a thin wrapper around the mysql library function [mysql_stmt_init_d\(\)](#).

Definition at line 52 of file `stmts.c`.

References `log_critical`, `mysql_stmt_init_d`, and `sql_error()`.

Referenced by `stmt_rebuild()`, and `stmt_start()`.

5.267.3.64 bool_t stmt_prepare (MYSQL_STMT * *group*, const char * *query*, unsigned long *length*)

Definition at line 116 of file `stmts.c`.

References `log_critical`, `mysql_stmt_attr_set_d`, `mysql_stmt_prepare_d`, and `stmt_error()`.

Referenced by `stmt_rebuild()`, and `stmt_start()`.

5.267.3.65 bool_t stmt_rebuild (uint32_t *connection*)

Definition at line 190 of file `stmts.c`.

References `log_critical`, `ns_length_get()`, `pool_get_obj()`, `queries`, `sql_pool`, `stmt_close()`, `stmt_open()`, and `stmt_prepare()`.

Referenced by `dspam_check()`, `dspam_train()`, and `stmt_reset()`.

5.267.3.66 MySQL_STMT* stmt_reset (MYSQL_STMT ** *group*, uint32_t *connection*)

Definition at line 222 of file stmts.c.

References log_critical, log_pedantic, mysql_stmt_reset_d, sql_ping(), and stmt_rebuild().

Referenced by stmt_exec_affected_conn(), stmt_exec_conn(), stmt_get_result_conn(), and stmt_insert_conn().

5.267.3.67 bool_t stmt_start (void)

Initialize the global array of mysql prepared statements.

Note:

This function readies a copy of each prepared statement for every member of the global mysql connection pool.

Returns:

true on success or false on failure.

Definition at line 143 of file stmts.c.

References magma_t::database, magma_t::iface, log_critical, magma, mm_alloc(), mm_wipe(), ns_length_get(), pool_get_obj(), queries, sql_pool, stmt_open(), stmt_prepare(), and stmt_stop().

Referenced by sql_start().

5.267.3.68 void stmt_stop (void)

Definition at line 90 of file stmts.c.

References magma_t::database, magma_t::iface, magma, mm_free(), queries, and stmt_close().

Referenced by sql_stop(), and stmt_start().

5.267.3.69 int64_t tran_commit (int64_t *transaction*)

[transaction.c](#) [transaction.c](#)

Parameters:

transaction the mysql connection identifier.

Returns:

0 on success, or a non-zero number on error.

Definition at line 72 of file transaction.c.

References command, length, log_info, PLACER, pool_get_obj(), pool_release(), sql_error(), sql_pool, sql_query_conn(), and tran_commands.

Referenced by api_endpoint_register(), mail_copy_message(), mail_move_message(), mail_remove_message(), mail_store_message(), meta_update_keys(), meta_update_realms(), portal_endpoint_alert_acknowledge(), register_business_step2(), tank_delete(), and tank_store().

5.267.3.70 int64_t tran_rollback (int64_t *transaction*)

Rollback a pending transaction in the mysql database.

Parameters:

transaction the mysql connection identifier.

Returns:

0 on success, or a non-zero number on error.

Definition at line 52 of file transaction.c.

References `command`, `length`, `log_info`, `PLACER`, `pool_get_obj()`, `pool_release()`, `sql_error()`, `sql_pool`, `sql_query_conn()`, and `tran_commands`.

Referenced by `api_endpoint_register()`, `mail_copy_message()`, `mail_move_message()`, `mail_remove_message()`, `mail_store_message()`, `meta_update_keys()`, `meta_update_realms()`, `portal_endpoint_alert_acknowledge()`, `register_business_step2()`, `tank_delete()`, and `tank_store()`.

5.267.3.71 int64_t tran_start (void)

Pull a connection from the SQL pool and start a transaction.

Returns:

-1 on failure, or a MySQL connection id from the SQL pool on success.

Definition at line 26 of file transaction.c.

References `command`, `length`, `log_info`, `PL_RESERVED`, `PLACER`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `sql_error()`, `sql_pool`, `sql_query_conn()`, and `tran_commands`.

Referenced by `api_endpoint_register()`, `mail_copy_message()`, `mail_move_message()`, `mail_remove_message()`, `mail_store_message()`, `meta_update_keys()`, `meta_update_realms()`, `portal_endpoint_alert_acknowledge()`, `register_business_step2()`, `tank_delete()`, and `tank_store()`.

5.267.4 Variable Documentation**5.267.4.1 pool_t* sql_pool**

Definition at line 143 of file mysql.c.

5.268 src/providers/database/mysql.c File Reference

```
#include "magma.h"
```

Functions

- [uint_t sql_errno](#) (MYSQL *mysql)
Get the last error number for a mysql connection.
- [const chr_t * sql_error](#) (MYSQL *mysql)
Get a human-readable error message for a mysql connection.
- [void sql_stop](#) (void)
Shutdown and free the pool of mysql connections.
- [MYSQL * sql_open](#) (bool_t silent)
Open up a new mysql connection to the configured database server.
- [int_t sql_ping](#) (uint32_t connection)
Ping the MySQL server via the provided connection. If the underlying TCP/IP socket has timed out, the ping function will will cause the MySQL client library to reestablish the connection. Because a new connection is created the collection of prepared statements will need to be reinitialized.
- [bool_t sql_start](#) (void)
Load up the mysql subsystem.
- [void sql_thread_stop](#) (void)
Prepare the thread for exiting to destroy mysql thread specific variables.
- [bool_t sql_thread_start](#) (void)
Initialize mysql thread specific variables for the calling thread.
- [const char * serv_type_mysql](#) (void)
Determine whether the mysql library in use is embedded or not.
- [const char * serv_charset_mysql](#) (void)
- [const char * serv_schema_mysql](#) (void)
- [const char * serv_version_mysql](#) (void)
Return the server version string of the mysql database.
- [const char * lib_version_mysql](#) (void)
Return the version string of libmysql.
- [bool_t lib_load_mysql](#) (void)
Initialize the libmysql and bind dynamically to the exported functions that are required.

Variables

- [pool_t * sql_pool](#) = NULL

- struct {
 char [lib_version](#) [16]
 char [serv_version](#) [16]
 char [serv_charset](#) [16]
 char [serv_schema](#) [32]
 const [chr_t](#) * [type_serv](#)
 const [chr_t](#) * [type_embed](#)
 const [chr_t](#) * [dash](#)
} [sql](#)

5.268.1 Function Documentation

5.268.1.1 [bool_t lib_load_mysql \(void\)](#)

Initialize the libmysql and bind dynamically to the exported functions that are required. [mysql.c](#)

Returns:

true on success or false on failure.

Definition at line 494 of file [mysql.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.268.1.2 [const char* lib_version_mysql \(void\)](#)

Return the version string of libmysql.

Returns:

a pointer to a character string containing the libmysql version information.

Definition at line 471 of file [mysql.c](#).

References [mysql_get_client_version_d](#), and [sql](#).

Referenced by [lib_load\(\)](#).

5.268.1.3 [const char* serv_charset_mysql \(void\)](#)

Definition at line 415 of file [mysql.c](#).

References [mm_wipe\(\)](#), [mysql_character_set_name_d](#), [mysql_close_d](#), [sql](#), and [sql_open\(\)](#).

Referenced by [lib_load\(\)](#).

5.268.1.4 [const char* serv_schema_mysql \(void\)](#)

Definition at line 431 of file [mysql.c](#).

References [mm_wipe\(\)](#), [mysql_close_d](#), [sql](#), and [sql_open\(\)](#).

Referenced by [lib_load\(\)](#).

5.268.1.5 `const char* serv_type_mysql (void)`

Determine whether the mysql library in use is embedded or not.

Returns:

`sql.type_embed` ("Embedded") or `sql.type_serv` ("MySQL");

Definition at line 401 of file `mysql.c`.

References `mysql_embedded_d`, and `sql`.

Referenced by `lib_load()`.

5.268.1.6 `const char* serv_version_mysql (void)`

Return the server version string of the mysql database.

Returns:

a pointer to a character string containing the mysql server version information.

Definition at line 451 of file `mysql.c`.

References `mm_wipe()`, `mysql_close_d`, `mysql_get_server_info_d`, `sql`, and `sql_open()`.

Referenced by `lib_load()`.

5.268.1.7 `uint_t sql_errno (MYSQL * mysql)`

Get the last error number for a mysql connection.

Parameters:

mysql a pointer to the MYSQL object of the connection to be queried.

Returns:

0 on failure, or the last mysql error message for the connection on success.

Definition at line 163 of file `mysql.c`.

References `mysql_errno_d`.

Referenced by `stmt_close()`.

5.268.1.8 `const chr_t* sql_error (MYSQL * mysql)`

Get a human-readable error message for a mysql connection.

Parameters:

mysql a pointer to the MYSQL object of the connection to be queried.

Returns:

NULL on failure, or a pointer to a null-terminated string with the error message on success.

Definition at line 177 of file `mysql.c`.

References `mysql_errno_d`, and `mysql_error_d`.

Referenced by `sql_open()`, `sql_ping()`, `sql_query_conn()`, `stmt_close()`, `stmt_open()`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

5.268.1.9 MYSQL* sql_open (bool_t *silent*)

Open up a new mysql connection to the configured database server.

Note:

The reconnect option will automatically be set on all new mysql connections.

Parameters:

silent if true, suppress logging of failure messages for this function.

Returns:

NULL on failure, or a pointer to a MYSQL objection for the newly established connection on success.

Definition at line 216 of file mysql.c.

References magma_t::database, magma_t::iface, log_critical, magma, mysql_close_d, mysql_init_d, mysql_options_d, mysql_real_connect_d, and sql_error().

Referenced by serv_charset_mysql(), serv_schema_mysql(), serv_version_mysql(), and sql_start().

5.268.1.10 int_t sql_ping (uint32_t *connection*)

Ping the MySQL server via the provided connection. If the underlying TCP/IP socket has timed out, the ping function will will cause the MySQL client library to reestablish the connection. Because a new connection is created the collection of prepared statements will need to be reinitialized.

Warning:

If the connection timed out, the ping will trigger a reconnect, which invalidates the prepared statement references.

```
{.c}
if (sql_ping(connection) < 0 || !stmt_rebuild(connection)) { log_error("Invalid
    database connection."); return; }
```

Parameters:

connection The specific connection inside the pool that should be used for the ping.

Returns:

Returns 1 if the library performed an automatic reconnect, 0 if the connection is active, and -1 if the reconnect failed.

Definition at line 296 of file mysql.c.

References log_error, mysql_ping_d, mysql_thread_id_d, pool_get_obj(), sql_error(), and sql_pool.

Referenced by dspam_check(), dspam_train(), and stmt_reset().

5.268.1.11 bool_t sql_start (void)

Load up the mysql subsystem.

Note:

This function will check that all necessary database parameters have been supplied, and that the supplied mysql library version is thread safe and initialized. It will also initialize the pool of database connections, and

Returns:

true on success or false on failure.

Definition at line 324 of file mysql.c.

References magma_t::database, magma_t::iface, log_critical, magma, mysql_server_init_d, mysql_thread_safe_d, ns_empty(), pool_alloc(), pool_set_obj(), sql_open(), sql_pool, sql_stop(), and stmt_start().

Referenced by process_start().

5.268.1.12 void sql_stop (void)

Shutdown and free the pool of mysql connections.

Returns:

This function returns no value.

Definition at line 189 of file mysql.c.

References magma_t::database, magma_t::iface, magma, my_once_free_d, mysql_close_d, mysql_server_end_d, pool_free(), pool_get_obj(), sql_pool, and stmt_stop().

Referenced by process_stop(), and sql_start().

5.268.1.13 bool_t sql_thread_start (void)

Initialize mysql thread specific variables for the calling thread.

Note:

mysql_thread_init()

Returns:

0 on success or non-zero on error.

Definition at line 388 of file mysql.c.

References log_error, and mysql_thread_init_d.

Referenced by thread_start().

5.268.1.14 void sql_thread_stop (void)

Prepare the thread for exiting to destroy mysql thread specific variables.

Note:

mysql_thread_end()

Returns:

This function returns no value.

Definition at line 378 of file mysql.c.

References mysql_thread_end_d.

Referenced by thread_stop().

5.268.2 Variable Documentation

5.268.2.1 const chr_t * dash

Definition at line 151 of file mysql.c.

5.268.2.2 char lib_version[16]

Definition at line 146 of file mysql.c.

5.268.2.3 char serv_charset[16]

Definition at line 148 of file mysql.c.

5.268.2.4 char serv_schema[32]

Definition at line 149 of file mysql.c.

5.268.2.5 char serv_version[16]

Definition at line 147 of file mysql.c.

5.268.2.6 struct { ... } sql

Referenced by lib_version_mysql(), serv_charset_mysql(), serv_schema_mysql(), serv_type_mysql(), and serv_version_mysql().

5.268.2.7 pool_t* sql_pool = NULL

Definition at line 143 of file mysql.c.

Referenced by dspam_check(), dspam_train(), mail_db_update_message_folder(), sql_insert(), sql_insert_conn(), sql_num_rows(), sql_num_rows_conn(), sql_ping(), sql_query(), sql_query_conn(), sql_query_res(), sql_query_res_conn(), sql_start(), sql_stop(), sql_write(), sql_write_conn(), stmt_exec(), stmt_exec_affected(), stmt_get_result(), stmt_insert(), stmt_rebuild(), stmt_start(), tran_commit(), tran_rollback(), and tran_start().

5.268.2.8 const chr_t * type_embed

Definition at line 151 of file mysql.c.

5.268.2.9 const chr_t* type_serv

Definition at line 151 of file mysql.c.

5.269 src/providers/database/query.c File Reference

```
#include "magma.h"
```

Functions

- MYSQL_RES * [sql_query_res_conn](#) ([stringer_t](#) *query, uint32_t connection)
- int64_t [sql_num_rows_conn](#) ([stringer_t](#) *query, uint32_t connection)
- int64_t [sql_insert_conn](#) ([stringer_t](#) *query, uint32_t connection)
- int64_t [sql_write_conn](#) ([stringer_t](#) *query, uint32_t connection)
- int64_t [sql_query_conn](#) ([stringer_t](#) *query, uint32_t connection)

Execute a mysql statement.

- MYSQL_RES * [sql_query_res](#) ([stringer_t](#) *query)
- int64_t [sql_num_rows](#) ([stringer_t](#) *query)
- int64_t [sql_insert](#) ([stringer_t](#) *query)

[query.c](#)

- int64_t [sql_write](#) ([stringer_t](#) *query)
- int64_t [sql_query](#) ([stringer_t](#) *query)

Pull a connection from the sql pool and execute a mysql statement.

5.269.1 Function Documentation

5.269.1.1 int64_t sql_insert (stringer_t * query)

[query.c](#)

Definition at line 124 of file [query.c](#).

References [log_pedantic](#), [PL_RESERVED](#), [pool_pull\(\)](#), [pool_release\(\)](#), [sql_insert_conn\(\)](#), and [sql_pool](#).

5.269.1.2 int64_t sql_insert_conn (stringer_t * query, uint32_t connection)

Definition at line 48 of file [query.c](#).

References [log_pedantic](#), [mysql_error_d](#), [mysql_insert_id_d](#), [mysql_real_query_d](#), [pool_get_obj\(\)](#), [sql_pool](#), [st_data_get\(\)](#), and [st_length_get\(\)](#).

Referenced by [sql_insert\(\)](#).

5.269.1.3 int64_t sql_num_rows (stringer_t * query)

Definition at line 109 of file [query.c](#).

References [log_pedantic](#), [PL_RESERVED](#), [pool_pull\(\)](#), [pool_release\(\)](#), [sql_num_rows_conn\(\)](#), and [sql_pool](#).

5.269.1.4 int64_t sql_num_rows_conn (stringer_t * query, uint32_t connection)

Definition at line 27 of file [query.c](#).

References [log_pedantic](#), [mysql_error_d](#), [mysql_free_result_d](#), [mysql_num_rows_d](#), [mysql_real_query_d](#), [mysql_store_result_d](#), [pool_get_obj\(\)](#), [sql_pool](#), [st_data_get\(\)](#), and [st_length_get\(\)](#).

Referenced by [sql_num_rows\(\)](#).

5.269.1.5 `int64_t sql_query (stringer_t * query)`

Pull a connection from the sql pool and execute a mysql statement.

See also:

[sql_query_conn\(\)](#)

Parameters:

query the mysql statement to be executed.

Returns:

0 on success, or non-zero on failure.

Definition at line 160 of file query.c.

References [log_info](#), [PL_RESERVED](#), [pool_pull\(\)](#), [pool_release\(\)](#), [sql_pool](#), and [sql_query_conn\(\)](#).

5.269.1.6 `int64_t sql_query_conn (stringer_t * query, uint32_t connection)`

Execute a mysql statement.

See also:

[mysql_real_query\(\)](#)

Parameters:

query the mysql statement to be executed.

connection a connection id for the underlying mysql session.

Returns:

0 on success, or a non-zero number on error.

Definition at line 83 of file query.c.

References [log_pedantic](#), [mysql_real_query_d](#), [pool_get_obj\(\)](#), [sql_error\(\)](#), [sql_pool](#), [st_char_get\(\)](#), [st_data_get\(\)](#), [st_length_get\(\)](#), and [st_length_int\(\)](#).

Referenced by [sql_query\(\)](#), [tran_commit\(\)](#), [tran_rollback\(\)](#), and [tran_start\(\)](#).

5.269.1.7 `MYSQL_RES* sql_query_res (stringer_t * query)`

Definition at line 94 of file query.c.

References [log_pedantic](#), [PL_RESERVED](#), [pool_pull\(\)](#), [pool_release\(\)](#), [sql_pool](#), and [sql_query_res_conn\(\)](#).

5.269.1.8 `MYSQL_RES* sql_query_res_conn (stringer_t * query, uint32_t connection)`

Definition at line 10 of file query.c.

References [log_pedantic](#), [mysql_error_d](#), [mysql_real_query_d](#), [mysql_store_result_d](#), [pool_get_obj\(\)](#), [sql_pool](#), [st_data_get\(\)](#), and [st_length_get\(\)](#).

Referenced by [sql_query_res\(\)](#).

5.269.1.9 int64_t sql_write (stringer_t * *query*)

Definition at line 139 of file query.c.

References log_pedantic, PL_RESERVED, pool_pull(), pool_release(), sql_pool, and sql_write_conn().

5.269.1.10 int64_t sql_write_conn (stringer_t * *query*, uint32_t *connection*)

Definition at line 62 of file query.c.

References log_pedantic, mysql_affected_rows_d, mysql_error_d, mysql_real_query_d, pool_get_obj(), sql_pool, st_data_get(), and st_length_get().

Referenced by sql_write().

5.270 src/providers/database/results.c File Reference

```
#include "magma.h"
```

Functions

- [row_t * res_field_generic](#) ([row_t](#) *row, [uint64_t](#) field, [size_t](#) typesize)
void * [res_field_block](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as generic pointer.
- [stringer_t * res_field_string](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as managed string.
- [double_t res_field_double](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as a double.
- [float_t res_field_float](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as a float.
- [uint64_t res_field_uint64](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as an unsigned 64-bit integer.
- [uint32_t res_field_uint32](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as an unsigned 32-bit integer.
- [uint16_t res_field_uint16](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as an unsigned 16-bit integer.
- [uint8_t res_field_uint8](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as an unsigned 8-bit integer.
- [int64_t res_field_int64](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as a signed 64-bit integer.
- [int32_t res_field_int32](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as a signed 32-bit integer.
- [int16_t res_field_int16](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as a signed 16-bit integer.
- [int8_t res_field_int8](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as a signed 8-bit integer.
- [bool_t res_field_bool](#) ([row_t](#) *row, [uint64_t](#) field)
Get the value of a specified field in a database result row as a boolean.
- [size_t res_field_length](#) ([row_t](#) *row, [uint64_t](#) field)
Get the length of a specified field in a database result row.
- void [res_table_free](#) ([table_t](#) *table)
Free a MySQL results table.

- void [res_bind_free](#) (MYSQL_STMT *stmt, MYSQL_BIND *binding, uint64_t *number*)
Free a prepared MySQL statement result set binding.
- [table_t](#) * [res_table_alloc](#) (uint64_t rows, uint64_t fields)
- uint64_t [res_bind_create](#) (MYSQL_STMT *stmt, MYSQL_BIND **result)
results.c
- uint64_t [res_field_count](#) ([table_t](#) *table)
Return the number of fields stored in the database results table.
- uint64_t [res_row_count](#) ([table_t](#) *table)
Return the number of rows in the MySQL results table.
- void [res_row_set](#) ([row_t](#) *row, [chr_t](#) *buffer)
Set the buffer location for a database result row.
- [row_t](#) * [res_row_get](#) ([table_t](#) *table, uint64_t row)
Retrieve a row from a database results table by index.
- [row_t](#) * [res_row_next](#) ([table_t](#) *table)
Return the next row in the database results table and advance the cursor.
- [bool_t](#) [res_row_store](#) (uint64_t num, [table_t](#) *table, MYSQL_BIND *binding)
- [table_t](#) * [res_stmt_store](#) (MYSQL_STMT *stmt)

5.270.1 Function Documentation

5.270.1.1 uint64_t res_bind_create (MYSQL_STMT * stmt, MYSQL_BIND ** result)

[results.c](#)

Definition at line 348 of file [results.c](#).

References [length](#), [log_info](#), [mm_alloc\(\)](#), [mysql_fetch_field_d](#), [mysql_free_result_d](#), [mysql_num_fields_d](#), [mysql_stmt_error_d](#), [mysql_stmt_result_metadata_d](#), and [res_bind_free\(\)](#).

Referenced by [res_stmt_store\(\)](#).

5.270.1.2 void res_bind_free (MYSQL_STMT * stmt, MYSQL_BIND * binding, uint64_t number)

Free a prepared MySQL statement result set binding.

Parameters:

- stmt* the input prepared MySQL statement.
- binding* the binding for the result set.
- number* the number of fields in the result set.

Returns:

This function does not return a value.

Definition at line 290 of file [results.c](#).

References [length](#), [mm_cleanup](#), [mm_free\(\)](#), and [mysql_stmt_bind_result_d](#).

Referenced by [res_bind_create\(\)](#), and [res_stmt_store\(\)](#).

5.270.1.3 void* res_field_block (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as generic pointer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a generic pointer, or NULL on failure.

Definition at line 43 of file results.c.

References res_field_generic().

Referenced by auth_data_fetch(), config_load_database_settings(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_fetch_alerts(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), meta_data_fetch_shard(), meta_data_fetch_user(), smtp_check_receive_quota(), smtp_fetch_inbound(), user_config_fetch(), and warehouse_fetch_domains().

5.270.1.4 bool_t res_field_bool (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as a boolean.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a boolean, or false on failure.

Definition at line 216 of file results.c.

References res_field_generic().

5.270.1.5 uint64_t res_field_count (table_t * table)

Return the number of fields stored in the database results table.

Parameters:

table the input database results table.

Returns:

0 on failure, or the number of table fields on success.

Definition at line 415 of file results.c.

References log_info.

Referenced by res_row_store().

5.270.1.6 `double_t res_field_double (row_t * row, uint64_t field)`

Get the value of a specified field in a database result row as a double.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a double, or 0 on failure.

Definition at line 86 of file results.c.

References `res_field_generic()`.

5.270.1.7 `float_t res_field_float (row_t * row, uint64_t field)`

Get the value of a specified field in a database result row as a float.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a float, or 0 on failure.

Definition at line 99 of file results.c.

References `res_field_generic()`.

5.270.1.8 `row_t* res_field_generic (row_t * row, uint64_t field, size_t typesize)`

Definition at line 10 of file results.c.

References `log_info`.

Referenced by `res_field_block()`, `res_field_bool()`, `res_field_double()`, `res_field_float()`, `res_field_int16()`, `res_field_int32()`, `res_field_int64()`, `res_field_int8()`, `res_field_uint16()`, `res_field_uint32()`, `res_field_uint64()`, and `res_field_uint8()`.

5.270.1.9 `int16_t res_field_int16 (row_t * row, uint64_t field)`

Get the value of a specified field in a database result row as a signed 16-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a signed 16-bit integer, or 0 on failure.

Definition at line 190 of file results.c.

References `res_field_generic()`.

5.270.1.10 int32_t res_field_int32 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as a signed 32-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a signed 32-bit integer, or 0 on failure.

Definition at line 177 of file results.c.

References res_field_generic().

5.270.1.11 int64_t res_field_int64 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as a signed 64-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a signed 64-bit integer, or 0 on failure.

Definition at line 164 of file results.c.

References res_field_generic().

5.270.1.12 int8_t res_field_int8 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as a signed 8-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a signed 8-bit integer, or 0 on failure.

Definition at line 203 of file results.c.

References res_field_generic().

Referenced by auth_data_fetch(), meta_data_fetch_user(), smtp_fetch_authorization(), smtp_fetch_inbound(), teacher_data_fetch(), and warehouse_fetch_domains().

5.270.1.13 size_t res_field_length (row_t * row, uint64_t field)

Get the length of a specified field in a database result row.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the length of the specified result row, or 0 on failure.

Definition at line 229 of file results.c.

References log_info.

Referenced by auth_data_fetch(), config_load_database_settings(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_fetch_alerts(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), meta_data_fetch_shard(), meta_data_fetch_user(), smtp_check_receive_quota(), smtp_fetch_inbound(), user_config_fetch(), and warehouse_fetch_domains().

5.270.1.14 stringer_t* res_field_string (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as managed string.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as a managed string, or NULL on failure.

Definition at line 56 of file results.c.

References length, log_info, and st_import().

Referenced by auth_data_fetch(), meta_data_fetch_all_tags(), meta_data_fetch_message_tags(), meta_data_fetch_user(), register_data_fetch_blocklist(), smtp_fetch_authorization(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_rollout(), teacher_data_fetch(), and warehouse_fetch_patterns().

5.270.1.15 uint16_t res_field_uint16 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as an unsigned 16-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as an unsigned 16-bit integer, or 0 on failure.

Definition at line 138 of file results.c.

References res_field_generic().

5.270.1.16 uint32_t res_field_uint32 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as an unsigned 32-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as an unsigned 32-bit integer, or 0 on failure.

Definition at line 125 of file results.c.

References res_field_generic().

Referenced by auth_data_fetch(), magma_folder_fetch(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_messages(), smtp_fetch_authorization(), smtp_fetch_inbound(), and smtp_rollout().

5.270.1.17 uint64_t res_field_uint64 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as an unsigned 64-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as an unsigned 64-bit integer, or 0 on failure.

Definition at line 112 of file results.c.

References res_field_generic().

Referenced by auth_data_fetch(), config_fetch_host_number(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_acknowledge_alert(), meta_data_fetch_alerts(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_messages(), smtp_check_receive_quota(), smtp_check_transmit_quota(), smtp_fetch_authorization(), smtp_fetch_inbound(), smtp_rollout(), statistics_refresh(), teacher_data_fetch(), and user_config_fetch().

5.270.1.18 uint8_t res_field_uint8 (row_t * row, uint64_t field)

Get the value of a specified field in a database result row as an unsigned 8-bit integer.

Parameters:

row a pointer to the database result row to be examined.

field the zero-based index of the field in the row to be queried.

Returns:

the value of the specified result row as an unsigned 8-bit integer, or 0 on failure.

Definition at line 151 of file results.c.

References res_field_generic().

Referenced by meta_data_fetch_mailbox_aliases(), and meta_data_fetch_shard().

5.270.1.19 uint64_t res_row_count (table_t * table)

Return the number of rows in the MySQL results table.

Note:

The row count is provided for a one-indexed table.

Parameters:

table the input MySQL results table.

Returns:

0 on error, or the row count on success.

Definition at line 432 of file results.c.

References log_info.

Referenced by auth_data_fetch(), config_load_database_settings(), meta_data_fetch_message_tags(), res_row_store(), smtp_check_authorized_from(), and smtp_fetch_inbound().

5.270.1.20 row_t* res_row_get (table_t * table, uint64_t row)

Retrieve a row from a database results table by index.

Note:

This function operates on a 0-indexed table.

Parameters:

table the input database results table. row the row # to be fetched.

Returns:

NULL on failure, or a pointer to a result row on success.

Definition at line 469 of file results.c.

References log_info.

Referenced by config_load_database_settings(), res_row_next(), and res_row_store().

5.270.1.21 row_t* res_row_next (table_t * table)

Return the next row in the database results table and advance the cursor.

Parameters:

table the input mysql results table.

Returns:

a pointer to the next result row in the table.

Definition at line 490 of file results.c.

References count, and res_row_get().

Referenced by auth_data_fetch(), config_fetch_host_number(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_acknowledge_alert(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(),

meta_data_fetch_keys(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_message_tags(), meta_data_fetch_messages(), meta_data_fetch_shard(), meta_data_fetch_user(), register_data_check_username(), register_data_fetch_blocklist(), smtp_check_receive_quota(), smtp_check_transmit_quota(), smtp_fetch_authorization(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_rollout(), statistics_refresh(), teacher_data_fetch(), user_config_fetch(), warehouse_fetch_domains(), and warehouse_fetch_patterns().

5.270.1.22 void res_row_set (row_t * row, chr_t * buffer)

Set the buffer location for a database result row.

Parameters:

row the input database result row.
buffer the buffer to be assigned to the target row.

Returns:

This function returns no value.

Definition at line 449 of file results.c.

References log_info.

Referenced by res_row_store().

5.270.1.23 bool_t res_row_store (uint64_t num, table_t * table, MYSQL_BIND * binding)

Definition at line 510 of file results.c.

References length, log_info, mm_alloc(), mm_copy(), res_field_count(), res_row_count(), res_row_get(), and res_row_set().

Referenced by res_stmt_store().

5.270.1.24 table_t* res_stmt_store (MYSQL_STMT * stmt)

Definition at line 579 of file results.c.

References log_info, mysql_stmt_bind_result_d, mysql_stmt_error_d, mysql_stmt_fetch_d, mysql_stmt_free_result_d, mysql_stmt_num_rows_d, mysql_stmt_store_result_d, res_bind_create(), res_bind_free(), res_row_store(), res_table_alloc(), and res_table_free().

Referenced by stmt_get_result_conn().

5.270.1.25 table_t* res_table_alloc (uint64_t rows, uint64_t fields)

Definition at line 316 of file results.c.

References log_info, and mm_alloc().

Referenced by res_stmt_store().

5.270.1.26 void res_table_free (table_t * table)

Free a MySQL results table.

Parameters:

table the MySQL results table to be freed.

Returns:

This function does not return a value.

Definition at line 250 of file results.c.

References `log_info`, and `mm_free()`.

Referenced by `auth_data_fetch()`, `config_fetch_host_number()`, `config_load_database_settings()`, `contact_details_fetch()`, `contacts_fetch()`, `magma_folder_fetch()`, `meta_data_acknowledge_alert()`, `meta_data_fetch_alerts()`, `meta_data_fetch_all_tags()`, `meta_data_fetch_folder_messages()`, `meta_data_fetch_folders()`, `meta_data_fetch_keys()`, `meta_data_fetch_mailbox_aliases()`, `meta_data_fetch_message_tags()`, `meta_data_fetch_messages()`, `meta_data_fetch_shard()`, `meta_data_fetch_user()`, `register_data_check_username()`, `register_data_fetch_blocklist()`, `res_stmt_store()`, `smtp_check_authorized_from()`, `smtp_check_receive_quota()`, `smtp_check_transmit_quota()`, `smtp_fetch_authorization()`, `smtp_fetch_autoreply()`, `smtp_fetch_inbound()`, `smtp_rollout()`, `statistics_refresh()`, `teacher_data_fetch()`, `user_config_fetch()`, `warehouse_fetch_domains()`, and `warehouse_fetch_patterns()`.

5.271 src/providers/database/stmts.c File Reference

```
#include "magma.h"
```

Functions

- [uint_t stmt_errno](#) (MYSQL_STMT *local)
Get the error number associated with a mysql statement.
- const [chr_t](#) * [stmt_error](#) (MYSQL_STMT *local)
- MYSQL_STMT * [stmt_open](#) (MYSQL *mysql)
Initialize a mysql prepared statement.
- void [stmt_close](#) (MYSQL_STMT *local)
- void [stmt_stop](#) (void)
- [bool_t](#) [stmt_prepare](#) (MYSQL_STMT *group, const char *query, unsigned long [length](#))
- [bool_t](#) [stmt_start](#) (void)
Initialize the global array of mysql prepared statements.
- [bool_t](#) [stmt_rebuild](#) (uint32_t connection)
- MYSQL_STMT * [stmt_reset](#) (MYSQL_STMT **group, uint32_t connection)
- [bool_t](#) [stmt_bind_param](#) (MYSQL_STMT *group, MYSQL_BIND *bind)
stmts.c
- [bool_t](#) [stmt_exec_conn](#) (MYSQL_STMT **group, MYSQL_BIND *parameters, uint32_t connection)
Execute a prepared mysql statement over a specified connection.
- [bool_t](#) [stmt_exec](#) (MYSQL_STMT **group, MYSQL_BIND *parameters)
Execute a prepared mysql statement.
- [table_t](#) * [stmt_get_result_conn](#) (MYSQL_STMT **group, MYSQL_BIND *parameters, uint32_t connection)
Execute a prepared mysql statement on a specified connection and return the result.
- [table_t](#) * [stmt_get_result](#) (MYSQL_STMT **group, MYSQL_BIND *parameters)
Execute a prepared mysql statement and return the result.
- uint64_t [stmt_insert_conn](#) (MYSQL_STMT **group, MYSQL_BIND *parameters, uint32_t connection)
Execute a mysql prepared INSERT or UPDATE statement on a specified connection.
- uint64_t [stmt_insert](#) (MYSQL_STMT **group, MYSQL_BIND *parameters)
Execute a mysql prepared INSERT or UPDATE statement.
- int64_t [stmt_exec_affected_conn](#) (MYSQL_STMT **group, MYSQL_BIND *parameters, uint32_t connection)
Execute a statement on a specified mysql connection and return the number of affected rows.
- int64_t [stmt_exec_affected](#) (MYSQL_STMT **group, MYSQL_BIND *parameters)
Execute a statement on a specified mysql connection and return the number of affected rows.

Variables

- [chr_t](#) * [queries](#) [] = { QUERIES_INIT }

5.271.1 Function Documentation

5.271.1.1 `bool_t stmt_bind_param (MYSQL_STMT * group, MYSQL_BIND * bind)`

[stmts.c](#)

Definition at line 251 of file `stmts.c`.

References `log_critical`, `log_pedantic`, `mysql_stmt_bind_param_d`, and `stmt_error()`.

Referenced by `stmt_exec_affected_conn()`, `stmt_exec_conn()`, `stmt_get_result_conn()`, and `stmt_insert_conn()`.

5.271.1.2 `void stmt_close (MYSQL_STMT * local)`

Definition at line 68 of file `stmts.c`.

References `log_critical`, `mysql_stmt_close_d`, `sql_errno()`, and `sql_error()`.

Referenced by `stmt_rebuild()`, and `stmt_stop()`.

5.271.1.3 `uint_t stmt_errno (MYSQL_STMT * local)`

Get the error number associated with a mysql statement.

Parameters:

local a pointer to the input mysql statement object.

Returns:

the errno associated with the specified mysql statement.

Definition at line 17 of file `stmts.c`.

References `mysql_errno_d`, and `mysql_stmt_errno_d`.

5.271.1.4 `const chr_t* stmt_error (MYSQL_STMT * local)`

Return the error string for a mysql statement.

Parameters:

local A single `MYSQL_STMT*` parameter.

Returns:

This function returns the mysql error string, or NULL if unable to retrieve it.

Definition at line 34 of file `stmts.c`.

References `mysql_errno_d`, `mysql_error_d`, `mysql_stmt_errno_d`, and `mysql_stmt_error_d`.

Referenced by `stmt_bind_param()`, `stmt_exec_affected_conn()`, `stmt_exec_conn()`, `stmt_get_result_conn()`, `stmt_insert_conn()`, and `stmt_prepare()`.

5.271.1.5 `bool_t stmt_exec (MYSQL_STMT ** group, MYSQL_BIND * parameters)`

Execute a prepared mysql statement.

See also:[stmt_exec_conn\(\)](#)**Parameters:***group* the mysql prepared statement to be executed.*parameters* the parameters to be bound to the statement.**Returns:**

false on failure, or true on success.

Definition at line 307 of file stmts.c.

References [log_info](#), [PL_RESERVED](#), [pool_pull\(\)](#), [pool_release\(\)](#), [sql_pool](#), and [stmt_exec_conn\(\)](#).Referenced by [mail_db_hide_message\(\)](#), [meta_data_flags_add\(\)](#), [meta_data_flags_remove\(\)](#), [meta_data_flags_replace\(\)](#), [smtp_update_receive_stats\(\)](#), [smtp_update_transmission_stats\(\)](#), and [teacher_data_delete\(\)](#).**5.271.1.6 int64_t stmt_exec_affected (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*)**

Execute a statement on a specified mysql connection and return the number of affected rows.

Note:

Technically the number of affected rows is a my_ulonglong value, but since we retrun -1 when an error occurs, we are using the signed equivalent, int64_t. This means if more than INT64_MAX (or 9,223,372,036,854,775,807) rows are ever altered, the function will return INT64_MAX (or 9,223,372,036,854,775,807) instead.

See also:[mysql_stmt_affected_rows\(\)](#)**Parameters:***group* the prepared mysql statement to be executed.*parameters* the parameters to be bound to the prepared statement.**Returns:**

-1 on failure, 0 if executed successfully, but no rows are affected, and a positive number with the number of altered rows upon success.

Definition at line 492 of file stmts.c.

References [log_info](#), [PL_RESERVED](#), [pool_pull\(\)](#), [pool_release\(\)](#), [sql_pool](#), and [stmt_exec_affected_conn\(\)](#).Referenced by [api_endpoint_delete_user\(\)](#), [auth_data_update_legacy\(\)](#), [auth_data_update_lock\(\)](#), [contact_delete\(\)](#), [contact_detail_delete\(\)](#), [contact_detail_upsert\(\)](#), [contact_update\(\)](#), [contact_update_stamp\(\)](#), [magma_folder_delete\(\)](#), [magma_folder_rename\(\)](#), [meta_data_delete_folder\(\)](#), [meta_data_delete_tag\(\)](#), [meta_data_insert_tag\(\)](#), [meta_data_truncate_tags\(\)](#), [meta_data_update_folder_name\(\)](#), [meta_data_update_log\(\)](#), [user_config_delete\(\)](#), and [user_config_upsert\(\)](#).**5.271.1.7 int64_t stmt_exec_affected_conn (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*, uint32_t *connection*)**

Execute a statement on a specified mysql connection and return the number of affected rows.

Note:

Technically the number of affected rows is a my_ulonglong value, but since we retrun -1 when an error occurs, we are using the signed equivalent, int64_t. This means if more than INT64_MAX (or 9,223,372,036,854,775,807) rows are ever altered, the function will return INT64_MAX (or 9,223,372,036,854,775,807) instead.

See also:

mysql_stmt_affected_rows()

Parameters:

group the prepared mysql statement to be executed.
parameters the parameters to be bound to the prepared statement.
connection the mysql connection id over which the specified statement will be executed.

Returns:

-1 on failure, 0 if executed successfully, but no rows are affected, and a positive number with the number of altered rows upon success.

Definition at line 440 of file stmts.c.

References log_info, mysql_stmt_affected_rows_d, mysql_stmt_execute_d, stmt_bind_param(), stmt_error(), stmt_reset(), and uint64_clamp().

Referenced by mail_db_delete_message(), mail_db_update_message_folder(), meta_data_acknowledge_alert(), meta_data_insert_keys(), meta_data_insert_shard(), stmt_exec_affected(), and tank_delete_object().

5.271.1.8 bool_t stmt_exec_conn (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*, uint32_t *connection*)

Execute a prepared mysql statement over a specified connection.

Note:

This function will also reset the prepared statement and bind its parameters.

Parameters:

group the mysql prepared statement to be executed.
parameters the parameters to be bound to the statement.
connection the mysql connection id.

Returns:

false on failure, or true on success.

Definition at line 278 of file stmts.c.

References log_info, mysql_stmt_execute_d, stmt_bind_param(), stmt_error(), and stmt_reset().

Referenced by mail_db_insert_duplicate_message(), mail_db_insert_message(), register_data_insert_user(), and stmt_exec().

5.271.1.9 table_t* stmt_get_result (MYSQL_STMT ** *group*, MYSQL_BIND * *parameters*)

Execute a prepared mysql statement and return the result.

Parameters:

group the prepared mysql statement to be executed.
parameters the parameters to be passed with the query.

Returns:

the result of the query, or NULL on failure.

Definition at line 357 of file stmts.c.

References log_info, PL_RESERVED, pool_pull(), pool_release(), sql_pool, and stmt_get_result_conn().

Referenced by auth_data_fetch(), config_fetch_host_number(), config_fetch_settings(), contact_details_fetch(), contacts_fetch(), magma_folder_fetch(), meta_data_fetch_alerts(), meta_data_fetch_all_tags(), meta_data_fetch_folder_messages(), meta_data_fetch_folders(), meta_data_fetch_mailbox_aliases(), meta_data_fetch_message_tags(), meta_data_fetch_messages(), meta_data_fetch_user(), register_data_check_username(), register_data_fetch_blocklist(), smtp_check_authorized_from(), smtp_check_receive_quota(), smtp_check_transmit_quota(), smtp_fetch_authorization(), smtp_fetch_autoreply(), smtp_fetch_inbound(), smtp_fetch_rollmessages(), statistics_refresh(), teacher_data_fetch(), user_config_fetch(), warehouse_fetch_domains(), and warehouse_fetch_patterns().

5.271.1.10 table_t* stmt_get_result_conn (MYSQL_STMT ** group, MYSQL_BIND * parameters, uint32_t connection)

Execute a prepared mysql statement on a specified connection and return the result.

Parameters:

group the prepared mysql statement to be executed.
parameters the parameters to be passed with the query.
connection the mysql connection identifier.

Returns:

the result of the query, or NULL on failure.

Definition at line 329 of file stmts.c.

References log_info, mysql_stmt_execute_d, res_stmt_store(), stmt_bind_param(), stmt_error(), and stmt_reset().

Referenced by meta_data_acknowledge_alert(), meta_data_fetch_keys(), meta_data_fetch_shard(), and stmt_get_result().

5.271.1.11 uint64_t stmt_insert (MYSQL_STMT ** group, MYSQL_BIND * parameters)

Execute a mysql prepared INSERT or UPDATE statement.

See also:

mysql_stmt_insert_id()

Parameters:

group the mysql prepared statement to be executed.
parameters the parameters to be bound to the prepared statement.

Returns:

0 on failure, or the last LAST_INSERT_ID() value returned for the mysql auto-increment column.

Definition at line 409 of file stmts.c.

References log_info, PL_RESERVED, pool_pull(), pool_release(), sql_pool, and stmt_insert_conn().

Referenced by contact_insert(), magma_folder_insert(), meta_data_insert_folder(), and smtp_insert_spamsig().

5.271.1.12 uint64_t stmt_insert_conn (MYSQL_STMT ** group, MYSQL_BIND * parameters, uint32_t connection)

Execute a mysql prepared INSERT or UPDATE statement on a specified connection.

See also:

mysql_stmt_insert_id()

Parameters:

group the mysql prepared statement to be executed.

parameters the parameters to be bound to the prepared statement.

connection the underlying connection on which the statement will be executed.

Returns:

0 on failure, or the last LAST_INSERT_ID() value returned for the mysql auto-increment column.

Definition at line 380 of file stmts.c.

References log_info, mysql_stmt_execute_d, mysql_stmt_insert_id_d, stmt_bind_param(), stmt_error(), and stmt_reset().

Referenced by mail_db_insert_duplicate_message(), mail_db_insert_message(), register_data_insert_user(), stmt_insert(), and tank_insert_object().

5.271.1.13 MYSQL_STMT* stmt_open (MYSQL * *mysql*)

Initialize a mysql prepared statement.

See also:

mysql_stmt_init()

Parameters:

mysql the underlying mysql connection.

Returns:

This function is a thin wrapper around the mysql library function [mysql_stmt_init_d\(\)](#).

Definition at line 52 of file stmts.c.

References log_critical, mysql_stmt_init_d, and sql_error().

Referenced by stmt_rebuild(), and stmt_start().

5.271.1.14 bool_t stmt_prepare (MYSQL_STMT * *group*, const char * *query*, unsigned long *length*)

Definition at line 116 of file stmts.c.

References log_critical, mysql_stmt_attr_set_d, mysql_stmt_prepare_d, and stmt_error().

Referenced by stmt_rebuild(), and stmt_start().

5.271.1.15 bool_t stmt_rebuild (uint32_t *connection*)

Definition at line 190 of file stmts.c.

References log_critical, ns_length_get(), pool_get_obj(), queries, sql_pool, stmt_close(), stmt_open(), and stmt_prepare().

Referenced by dspam_check(), dspam_train(), and stmt_reset().

5.271.1.16 MYSQL_STMT* stmt_reset (MYSQL_STMT ** *group*, uint32_t *connection*)

Definition at line 222 of file stmts.c.

References log_critical, log_pedantic, mysql_stmt_reset_d, sql_ping(), and stmt_rebuild().

Referenced by stmt_exec_affected_conn(), stmt_exec_conn(), stmt_get_result_conn(), and stmt_insert_conn().

5.271.1.17 `bool_t stmt_start (void)`

Initialize the global array of mysql prepared statements.

Note:

This function readies a copy of each prepared statement for every member of the global mysql connection pool.

Returns:

true on success or false on failure.

Definition at line 143 of file stmts.c.

References magma_t::database, magma_t::iface, log_critical, magma, mm_alloc(), mm_wipe(), ns_length_get(), pool_get_obj(), queries, sql_pool, stmt_open(), stmt_prepare(), and stmt_stop().

Referenced by sql_start().

5.271.1.18 `void stmt_stop (void)`

Definition at line 90 of file stmts.c.

References magma_t::database, magma_t::iface, magma, mm_free(), queries, and stmt_close().

Referenced by sql_stop(), and stmt_start().

5.271.2 Variable Documentation

5.271.2.1 `chr_t* queries[] = { QUERIES_INIT }`

Definition at line 10 of file stmts.c.

Referenced by sanity_check(), stmt_rebuild(), stmt_start(), and stmt_stop().

5.272 src/providers/database/transaction.c File Reference

```
#include "magma.h"
```

Functions

- `int64_t tran_start (void)`
Pull a connection from the SQL pool and start a transaction.
- `int64_t tran_rollback (int64_t transaction)`
Rollback a pending transaction in the mysql database.
- `int64_t tran_commit (int64_t transaction)`
Commit a transaction to the mysql database.

Variables

- struct {
 `chr_t * command`
 `size_t length`
} `tran_commands` [3]

5.272.1 Function Documentation

5.272.1.1 `int64_t tran_commit (int64_t transaction)`

Commit a transaction to the mysql database. [transaction.c](#)

Parameters:

transaction the mysql connection identifier.

Returns:

0 on success, or a non-zero number on error.

Definition at line 72 of file `transaction.c`.

References `command`, `length`, `log_info`, `PLACER`, `pool_get_obj()`, `pool_release()`, `sql_error()`, `sql_pool`, `sql_query_conn()`, and `tran_commands`.

Referenced by `api_endpoint_register()`, `mail_copy_message()`, `mail_move_message()`, `mail_remove_message()`, `mail_store_message()`, `meta_update_keys()`, `meta_update_realms()`, `portal_endpoint_alert_acknowledge()`, `register_business_step2()`, `tank_delete()`, and `tank_store()`.

5.272.1.2 `int64_t tran_rollback (int64_t transaction)`

Rollback a pending transaction in the mysql database.

Parameters:

transaction the mysql connection identifier.

Returns:

0 on success, or a non-zero number on error.

Definition at line 52 of file transaction.c.

References `command`, `length`, `log_info`, `PLACER`, `pool_get_obj()`, `pool_release()`, `sql_error()`, `sql_pool`, `sql_query_conn()`, and `tran_commands`.

Referenced by `api_endpoint_register()`, `mail_copy_message()`, `mail_move_message()`, `mail_remove_message()`, `mail_store_message()`, `meta_update_keys()`, `meta_update_realms()`, `portal_endpoint_alert_acknowledge()`, `register_business_step2()`, `tank_delete()`, and `tank_store()`.

5.272.1.3 int64_t tran_start (void)

Pull a connection from the SQL pool and start a transaction.

Returns:

-1 on failure, or a MySQL connection id from the SQL pool on success.

Definition at line 26 of file transaction.c.

References `command`, `length`, `log_info`, `PL_RESERVED`, `PLACER`, `pool_get_obj()`, `pool_pull()`, `pool_release()`, `sql_error()`, `sql_pool`, `sql_query_conn()`, and `tran_commands`.

Referenced by `api_endpoint_register()`, `mail_copy_message()`, `mail_move_message()`, `mail_remove_message()`, `mail_store_message()`, `meta_update_keys()`, `meta_update_realms()`, `portal_endpoint_alert_acknowledge()`, `register_business_step2()`, `tank_delete()`, and `tank_store()`.

5.272.2 Variable Documentation**5.272.2.1 chr_t* command**

Definition at line 14 of file transaction.c.

Referenced by `__attribute__()`, `dntp_process()`, `imap_process()`, `molten_parse()`, `pop_process()`, `portal_endpoint()`, `smtp_process()`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

5.272.2.2 size_t length

Definition at line 15 of file transaction.c.

5.272.2.3 struct { ... } tran_commands[3]

LOW: Standardize how SQL connection and transaction handles are passed around. Some functions use `int64_t`, others use `uint32_t`. Perhaps we need to use a type def? Or perhaps create a similar type mapping for pool object handles?

Referenced by `tran_commit()`, `tran_rollback()`, and `tran_start()`.

5.273 src/providers/deprecated/deprecated.h File Reference

Functions

- uint64_t [cryptex_body_length](#) (cryptex_t *cryptex)
cryptex.c
- uint64_t [cryptex_envelope_length](#) (cryptex_t *cryptex)
Get the length of a cryptex envelope.
- uint64_t [cryptex_hmac_length](#) (cryptex_t *cryptex)
Get the length of a cryptex HMAC.
- uint64_t [cryptex_original_length](#) (cryptex_t *cryptex)
Get the original length of a cryptex object's data buffer.
- uint64_t [cryptex_total_length](#) (cryptex_t *cryptex)
Determine the total length of the data underlying a cryptex object.
- void * [cryptex_alloc](#) (uint64_t envelope, uint64_t hmac, uint64_t original, uint64_t body)
Allocate a new cryptex object.
- void * [cryptex_body_data](#) (cryptex_t *cryptex)
Get a pointer to the encrypted data body of a cryptex object.
- void * [cryptex_envelope_data](#) (cryptex_t *cryptex)
Get a pointer to the envelope data of a cryptex object.
- void * [cryptex_hmac_data](#) (cryptex_t *cryptex)
Get a pointer to the HMAC data of a cryptex object.
- void [cryptex_free](#) (cryptex_t *cryptex)
Free a cryptex object.
- uchr_t * [ecies_decrypt](#) (stringer_t *key, ECIES_KEY_TYPE key_type, cryptex_t *cryptex, size_t *length)
ecies.c
- cryptex_t * [ecies_encrypt](#) (stringer_t *key, ECIES_KEY_TYPE key_type, unsigned char *data, size_t length)
Encrypt a block of data using an ECIES public key.
- void * [ecies_envelope_derivation](#) (const void *input, size_t ilen, void *output, size_t *olen)
- EC_GROUP * [ecies_group](#) (uint64_t curve, bool_t precompute)
- EC_KEY * [ecies_key_alloc](#) (void)
Allocate a new ECIES key pair from the curve defined by ECIES_CURVE.
- EC_KEY * [ecies_key_create](#) (void)
Generate a random ECIES key pair.
- void [ecies_key_free](#) (EC_KEY *key)
Free an ECIES key pair.
- EC_KEY * [ecies_key_private](#) (uint64_t format, placer_t data)
- uchr_t * [ecies_key_private_bin](#) (EC_KEY *key, size_t *olen)

Return an ECIES private key as binary data.

- `stringer_t * ecies_key_private_hex` (`EC_KEY *key`)

Return an ECIES private key as a hex string.

- `EC_KEY * ecies_key_public` (`uint64_t format`, `placer_t data`)
- `uchr_t * ecies_key_public_bin` (`EC_KEY *key`, `size_t *olen`)

Return an ECIES public key as binary data.

- `stringer_t * ecies_key_public_hex` (`EC_KEY *key`)

Return an ECIES public key as a null-terminated hex string.

- `bool_t ecies_start` (`void`)
- `void ecies_stop` (`void`)

Destroy all initialized ECIES curve group information.

- `stringer_t * hmac_digest` (`digest_t *digest`, `stringer_t *s`, `stringer_t *key`, `stringer_t *output`)

hmac.c

- `stringer_t * hmac_md4` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * hmac_md5` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * hmac_ripemd160` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * hmac_sha` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * hmac_sha1` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * hmac_sha224` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * hmac_sha256` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * hmac_sha384` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `stringer_t * hmac_sha512` (`stringer_t *s`, `stringer_t *key`, `stringer_t *output`)
- `scramble_t * scramble_alloc` (`size_t length`)

scramble.c

- `void * scramble_body_data` (`scramble_t *buffer`)

Get a pointer to the start of the encrypted data from a scrambled data header.

- `uint64_t scramble_body_hash` (`scramble_t *buffer`)

Get a hash of a scrambled object's (encrypted) body data.

- `uint64_t scramble_body_length` (`scramble_t *buffer`)

Get the length of a scrambled object's (encrypted) body data.

- `stringer_t * scramble_decrypt` (`stringer_t *key`, `scramble_t *input`)

Un-scramble a block of data using an decryption key.

- `scramble_t * scramble_encrypt` (`stringer_t *key`, `stringer_t *input`)

Scramble a block of data using an encryption key.

- `void scramble_free` (`scramble_t *buffer`)

Free a scrambled data block.

- `void scramble_cleanup` (`scramble_t *buffer`)

Performed a checked free of a scramble buffer.

- `scramble_t * scramble_import` (`stringer_t *s`)

Return a managed string as a scrambled buffer, after validation.

- `uint64_t scramble_orig_length (scramble_t *buffer)`
Get the length of the original (unencrypted) data underlying a scrambled object.
- `uint64_t scramble_total_length (scramble_t *buffer)`
Get the total length of a scrambled object.
- `void * scramble_vector_data (scramble_t *buffer)`
Get a pointer to the start of the IV from a scrambled data header.
- `uint64_t scramble_vector_length (scramble_t *buffer)`
Get the length of a scrambled object's IV.
- `stringer_t * symmetric_decrypt (cipher_t *cipher, stringer_t *vector, stringer_t *key, stringer_t *input)`
symmetric.c
- `stringer_t * symmetric_encrypt (cipher_t *cipher, stringer_t *vector, stringer_t *key, stringer_t *input)`
HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.
- `stringer_t * symmetric_key (cipher_t *cipher, stringer_t *key, stringer_t *output)`
Derive a symmetric key from a user's password.
- `stringer_t * symmetric_vector (cipher_t *cipher, stringer_t *output)`
Generate an IV data suitable for the specified cipher.

5.273.1 Function Documentation

5.273.1.1 `void* cryptex_alloc (uint64_t envelope, uint64_t hmac, uint64_t original, uint64_t body)`

Allocate a new cryptex object.

Definition at line 109 of file cryptex.c.

References `cryptex_head_t`, `log_pedantic`, and `mm_alloc()`.

Referenced by `ecies_encrypt()`.

5.273.1.2 `void* cryptex_body_data (cryptex_t * cryptex)`

Get a pointer to the encrypted data body of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's encrypted data body.

Definition at line 98 of file cryptex.c.

References `cryptex_head_t`.

Referenced by `ecies_decrypt()`, and `ecies_encrypt()`.

5.273.1.3 uint64_t cryptex_body_length (cryptex_t * cryptex)

cryptex.c cryptex.c

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex encrypted data body.

Definition at line 40 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.273.1.4 void* cryptex_envelope_data (cryptex_t * cryptex)

Get a pointer to the envelope data of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's envelope data.

Definition at line 76 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.273.1.5 uint64_t cryptex_envelope_length (cryptex_t * cryptex)

Get the length of a cryptex envelope.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex envelope.

Definition at line 16 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt().

5.273.1.6 void cryptex_free (cryptex_t * cryptex)

Free a cryptex object.

Parameters:

cryptex the cryptex object to be freed.

Returns:

This function returns no value.

Definition at line 133 of file cryptex.c.

Referenced by ecies_encrypt().

5.273.1.7 void* cryptex_hmac_data (cryptex_t * cryptex)

Get a pointer to the HMAC data of a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

a pointer to the cryptex object's hmac data.

Definition at line 86 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.273.1.8 uint64_t cryptex_hmac_length (cryptex_t * cryptex)

Get the length of a cryptex HMAC.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex HMAC.

Definition at line 28 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.273.1.9 uint64_t cryptex_original_length (cryptex_t * cryptex)

Get the original length of a cryptex object's data buffer.

Parameters:

cryptex a pointer to the head of the cryptex object to be examined.

Returns:

the length, in bytes, of the cryptex object's original data.

Definition at line 52 of file cryptex.c.

References cryptex_head_t.

Referenced by ecies_decrypt().

5.273.1.10 uint64_t cryptex_total_length (cryptex_t * cryptex)

Determine the total length of the data underlying a cryptex object.

Parameters:

cryptex the input cryptex object.

Returns:

the total length of the associated cryptex data in bytes.

Definition at line 64 of file cryptex.c.

References cryptex_head_t.

5.273.1.11 uchr_t* ecies_decrypt (stringer_t * key, ECIES_KEY_TYPE key_type, cryptex_t * cryptex, size_t * length)

ecies.c ecies.c

Parameters:

key the ECIES private key in the specified format.

key_type the encoding type of the ECIES private key (ECIES_PRIVATE_BINARY or ECIES_PRIVATE_HEX).

cryptex a pointer to the head of the cryptex object with the encrypted data.

length a pointer to a size_t variable which will receive the final length of the unencrypted data.

Returns:

NULL on failure, or a pointer to a memory address containing the decrypted data on success..

Definition at line 579 of file ecies.c.

References cryptex_body_data(), cryptex_body_length(), cryptex_envelope_data(), cryptex_envelope_length(), cryptex_hmac_data(), cryptex_hmac_length(), cryptex_original_length(), EC_KEY_free_d, EC_KEY_get0_public_key_d, ECDH_compute_key_d, ECIES_CIPHER, ecies_envelope_derivation(), ECIES_HMAC, ecies_key_private(), ecies_key_public(), ECIES_PRIVATE_BINARY, ECIES_PRIVATE_HEX, ECIES_PUBLIC_BINARY, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_set_padding_d, EVP_CIPHER_key_length_d, EVP_DecryptFinal_ex_d, EVP_DecryptInit_ex_d, EVP_DecryptUpdate_d, EVP_get_cipherbyname_d, EVP_get_digestbyname_d, HMAC_CTX_cleanup_d, HMAC_CTX_init_d, HMAC_Final_d, HMAC_Init_ex_d, HMAC_Update_d, log_info, MEMORYBUF, mm_alloc(), OBJ_nid2sn_d, pl_init(), ssl_error_string(), and st_empty_out().

5.273.1.12 cryptex_t* ecies_encrypt (stringer_t * key, ECIES_KEY_TYPE key_type, unsigned char * data, size_t length)

Encrypt a block of data using an ECIES public key.

Parameters:

key the ECIES public key in the specified format.

key_type the encoding type of the ECIES public key (ECIES_PUBLIC_BINARY or ECIES_PUBLIC_HEX).

data a pointer to the block of data to be encrypted.

length the length, in bytes, of the data to be encrypted.

Returns:

NULL on failure, or a pointer to the header of the cryptex object containing the encrypted data on success..

Definition at line 408 of file ecies.c.

References cryptex_alloc(), cryptex_body_data(), cryptex_body_length(), cryptex_envelope_data(), cryptex_free(), cryptex_hmac_data(), cryptex_hmac_length(), EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2oct_d, ECDH_compute_key_d, ECIES_CIPHER, ecies_envelope_derivation(), ECIES_HMAC, ecies_key_create(), ecies_key_public(), ECIES_PUBLIC_BINARY, ECIES_PUBLIC_HEX, EVP_CIPHER_block_size_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_set_padding_d, EVP_CIPHER_key_length_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, EVP_get_cipherbyname_d, EVP_get_digestbyname_d, EVP_MD_size_d, HMAC_CTX_cleanup_d, HMAC_CTX_init_d, HMAC_Final_d, HMAC_Init_ex_d, HMAC_Update_d, log_info, MEMORYBUF, OBJ_nid2sn_d, pl_init(), ssl_error_string(), and st_empty_out().

5.273.1.13 void* ecies_envelope_derivation (const void *input, size_t ilen, void *output, size_t *olen)

Definition at line 391 of file ecies.c.

References ecies_envelope_evp, and EVP_Digest_d.

Referenced by ecies_decrypt(), and ecies_encrypt().

5.273.1.14 EC_GROUP* ecies_group (uint64_t curve, bool_t precompute)

Definition at line 90 of file ecies.c.

References EC_GROUP_free_d, EC_GROUP_new_by_curve_name_d, EC_GROUP_precompute_mult_d, EC_GROUP_set_point_conversion_form_d, log_error, MEMORYBUF, and ssl_error_string().

Referenced by ecies_start().

5.273.1.15 EC_KEY* ecies_key_alloc (void)

Allocate a new ECIES key pair from the curve defined by ECIES_CURVE.

See also:

NID_sect571k1

Returns:

NULL on failure, or a pointer to the newly allocated key pair.

Definition at line 59 of file ecies.c.

References EC_KEY_free_d, EC_KEY_new_by_curve_name_d, EC_KEY_new_d, EC_KEY_set_conv_form_d, EC_KEY_set_group_d, ECIES_CURVE, ecies_curve_group, log_info, MEMORYBUF, and ssl_error_string().

Referenced by ecies_key_create(), ecies_key_private(), and ecies_key_public().

5.273.1.16 EC_KEY* ecies_key_create (void)

Generate a random ECIES key pair.

Returns:

NULL on failure, or a new random ECIES key pair on success.

Definition at line 113 of file ecies.c.

References EC_KEY_free_d, EC_KEY_generate_key_d, ecies_key_alloc(), log_info, MEMORYBUF, and ssl_error_string().

Referenced by ecies_encrypt().

5.273.1.17 void ecies_key_free (EC_KEY * *key*)

Free an ECIES key pair.

Returns:

This function returns no value.

Definition at line 48 of file ecies.c.

References EC_KEY_free_d.

5.273.1.18 EC_KEY* ecies_key_private (uint64_t *format*, placer_t *data*)

Definition at line 200 of file ecies.c.

References BN_bin2bn_d, BN_free_d, BN_hex2bn_d, EC_KEY_free_d, EC_KEY_set_private_key_d, ecies_key_alloc(), ECIES_PRIVATE_BINARY, ECIES_PRIVATE_HEX, log_info, MEMORYBUF, number, pl_char_get(), pl_data_get(), pl_length_get(), and ssl_error_string().

Referenced by ecies_decrypt().

5.273.1.19 uchr_t* ecies_key_private_bin (EC_KEY * *key*, size_t * *olen*)

Return an ECIES private key as binary data.

Parameters:

key the input ECIES key pair.

olen a pointer to store the length of the returned key.

Returns:

NULL on failure, or a pointer to the raw private key.

Definition at line 360 of file ecies.c.

References BN_bn2bin_d, BN_num_bytes_d, EC_KEY_get0_private_key_d, log_info, MEMORYBUF, mm_sec_alloc(), mm_sec_free(), and ssl_error_string().

5.273.1.20 stringer_t* ecies_key_private_hex (EC_KEY * *key*)

Return an ECIES private key as a hex string.

Parameters:

key the input ECIES key pair.

Returns:

NULL on failure, or the hex-formatted private key as a managed string.

Definition at line 326 of file ecies.c.

References BN_bn2hex_d, CONTIGUOUS, EC_KEY_get0_private_key_d, log_pedantic, MANAGED_T, MEMORYBUF, ns_length_get(), ns_wipe(), OPENSSL_free_d, SECURE, ssl_error_string(), and st_import_opts().

5.273.1.21 EC_KEY* ecies_key_public (uint64_t *format*, placer_t *data*)

Definition at line 132 of file ecies.c.

References EC_KEY_check_key_d, EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_set_public_key_d, EC_POINT_free_d, EC_POINT_hex2point_d, EC_POINT_new_d, EC_POINT_oct2point_d, ecies_key_alloc(), ECIES_PUBLIC_BINARY, ECIES_PUBLIC_HEX, log_info, MEMORYBUF, pl_char_get(), pl_data_get(), pl_length_get(), and ssl_error_string().

Referenced by ecies_decrypt(), and ecies_encrypt().

5.273.1.22 uchr_t* ecies_key_public_bin (EC_KEY * *key*, size_t * *olen*)

Return an ECIES public key as binary data.

Parameters:

key the input ECIES key pair.

olen a pointer to store the length of the returned key.

Returns:

NULL on failure, or a pointer to the raw public key.

Definition at line 289 of file ecies.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2oct_d, log_info, MEMORYBUF, mm_alloc(), mm_free(), and ssl_error_string().

5.273.1.23 stringer_t* ecies_key_public_hex (EC_KEY * *key*)

Return an ECIES public key as a null-terminated hex string.

Parameters:

key the input ECIES key pair.

Returns:

NULL on failure, or the hex-formatted public key as a null-terminated string.

Definition at line 256 of file ecies.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_POINT_point2hex_d, log_info, MEMORYBUF, ns_length_get(), OPENSSL_free_d, ssl_error_string(), and st_import().

5.273.1.24 bool_t ecies_start (void)

Definition at line 16 of file ecies.c.

References ECIES_CIPHER, ecies_cipher_evp, ECIES_CURVE, ecies_curve_group, ECIES_ENVELOPE, ecies_envelope_evp, ecies_group(), ECIES_HMAC, ecies_hmac_evp, EVP_get_cipherbyname_d, EVP_get_digestbyname_d, log_error, and OBJ_nid2sn_d.

5.273.1.25 void ecies_stop (void)

Destroy all initialized ECIES curve group information.

Returns:

This function returns no value.

Definition at line 31 of file ecies.c.

References EC_GROUP_free_d, and ecies_curve_group.

5.273.1.26 stringer_t* hmac_digest (digest_t * *digest*, stringer_t * *s*, stringer_t * *key*, stringer_t * *output*)

hmac.c hmac.c

Parameters:

digest Digest to be used with the HMAC.

s Input data.

key Key used in HMAC.

output Stringer containing buffer for output.

Returns:

The managed string containing the resulting HMAC or NULL if an error occurs.

Definition at line 21 of file hmac.c.

References EVP_MD_size_d, HMAC_CTX_cleanup_d, HMAC_CTX_init_d, HMAC_Final_d, HMAC_Init_ex_d, HMAC_Update_d, log_pedantic, st_alloc(), st_avail_get(), st_data_get(), st_empty, st_free(), st_length_get(), st_length_set(), st_valid_avail(), st_valid_destination(), and st_valid_tracked().

Referenced by hmac_md4(), hmac_md5(), hmac_ripemd160(), hmac_sha(), hmac_sha1(), hmac_sha224(), hmac_sha256(), hmac_sha384(), and hmac_sha512().

5.273.1.27 stringer_t* hmac_md4 (stringer_t * *s*, stringer_t * *key*, stringer_t * *output*)

Definition at line 91 of file hmac.c.

References EVP_md4_d, and hmac_digest().

5.273.1.28 stringer_t* hmac_md5 (stringer_t * *s*, stringer_t * *key*, stringer_t * *output*)

Definition at line 95 of file hmac.c.

References EVP_md5_d, and hmac_digest().

5.273.1.29 stringer_t* hmac_ripemd160 (stringer_t * *s*, stringer_t * *key*, stringer_t * *output*)

Definition at line 123 of file hmac.c.

References EVP_ripemd160_d, and hmac_digest().

5.273.1.30 stringer_t* hmac_sha (stringer_t * *s*, stringer_t * *key*, stringer_t * *output*)

Definition at line 99 of file hmac.c.

References EVP_sha_d, and hmac_digest().

5.273.1.31 stringer_t* hmac_sha1 (stringer_t * *s*, stringer_t * *key*, stringer_t * *output*)

Definition at line 103 of file hmac.c.

References EVP_sha1_d, and hmac_digest().

5.273.1.32 stringer_t* hmac_sha224 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 107 of file hmac.c.

References EVP_sha224_d, and hmac_digest().

5.273.1.33 stringer_t* hmac_sha256 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 111 of file hmac.c.

References EVP_sha256_d, and hmac_digest().

5.273.1.34 stringer_t* hmac_sha384 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 115 of file hmac.c.

References EVP_sha384_d, and hmac_digest().

5.273.1.35 stringer_t* hmac_sha512 (stringer_t * s, stringer_t * key, stringer_t * output)

Definition at line 119 of file hmac.c.

References EVP_sha512_d, and hmac_digest().

5.273.1.36 scramble_t* scramble_alloc (size_t length)

scramble.c scramble.c

Parameters:

length the length, in bytes, of the scrambled data buffer (should include the IV and encrypted body length).

Returns:

a pointer to a newly allocated scrambled data header.

Definition at line 143 of file scramble.c.

References mm_alloc(), and scramble_head_t.

Referenced by scramble_encrypt().

5.273.1.37 void* scramble_body_data (scramble_t * buffer)

Get a pointer to the start of the encrypted data from a scrambled data header.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

a pointer to the start of the scrambled data header's associated encrypted data body.

Definition at line 82 of file scramble.c.

References scramble_head_t, and scramble_vector_length().

Referenced by scramble_decrypt(), scramble_encrypt(), and scramble_import().

5.273.1.38 uint64_t scramble_body_hash (scramble_t * *buffer*)

Get a hash of a scrambled object's (encrypted) body data.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the 64-bit hash of the specified scrambled object's body data.

Definition at line 34 of file scramble.c.

References `scramble_head_t`.

5.273.1.39 uint64_t scramble_body_length (scramble_t * *buffer*)

Get the length of a scrambled object's (encrypted) body data.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's body data.

Definition at line 58 of file scramble.c.

References `scramble_head_t`.

Referenced by `scramble_decrypt()`, and `scramble_total_length()`.

5.273.1.40 void scramble_cleanup (scramble_t * *buffer*)

Performed a checked free of a scramble buffer.

See also:

[scramble_free](#)

Parameters:

block the scramble buffer to be freed.

Returns:

This function returns no value.

Definition at line 166 of file scramble.c.

References `mm_free()`.

5.273.1.41 stringer_t* scramble_decrypt (stringer_t * *key*, scramble_t * *input*)

Un-scramble a block of data using an decryption key.

Parameters:

key a managed string containing the symmetric decryption key.

input a pointer to the scrambled data header to be decrypted.

Returns:

NULL on failure, or a managed string containing the verified decrypted data.

Definition at line 239 of file scramble.c.

References cipher_id(), hash_adler32(), log_info, log_pedantic, MANAGEDBUF, PLACER, scramble_body_data(), scramble_body_length(), scramble_head_t, scramble_vector_data(), scramble_vector_length(), st_free(), st_length_get(), symmetric_decrypt(), and symmetric_key().

5.273.1.42 **scramble_t* scramble_encrypt (stringer_t * *key*, stringer_t * *input*)**

Scramble a block of data using an encryption key.

Note:

This function is configured to use AES 256 in CBC mode.

Parameters:

key a managed string containing the symmetric encryption key.

input a managed string containing the data to be encrypted.

Returns:

NULL on failure, or a pointer to a scrambled data header containing the encrypted data and its metadata.

Definition at line 182 of file scramble.c.

References cipher_id(), cipher_numeric_id(), hash_adler32(), log_pedantic, MANAGEDBUF, mm_copy(), scramble_alloc(), scramble_body_data(), scramble_head_t, scramble_vector_data(), st_data_get(), st_free(), st_length_get(), symmetric_encrypt(), symmetric_key(), and symmetric_vector().

5.273.1.43 **void scramble_free (scramble_t * *buffer*)**

Free a scrambled data block.

Parameters:

buffer a pointer to the scrambled data header to be freed.

Returns:

This function returns no value.

Definition at line 153 of file scramble.c.

References mm_free().

5.273.1.44 **scramble_t* scramble_import (stringer_t * *s*)**

Return a managed string as a scrambled buffer, after validation.

Note:

The returned pointer is inside the existing stringer, so it doesn't need be freed.

Parameters:

s a managed string containing the serialized scrambled data.

Returns:

NULL on failure, or a pointer to the scrambled data header on success.

Definition at line 103 of file scramble.c.

References hash_adler32(), log_pedantic, scramble_body_data(), scramble_head_t, st_data_get(), st_empty, and st_length_get().

5.273.1.45 uint64_t scramble_orig_length (scramble_t * *buffer*)

Get the length of the original (unencrypted) data underlying a scrambled object.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's original data.

Definition at line 46 of file scramble.c.

References scramble_head_t.

5.273.1.46 uint64_t scramble_total_length (scramble_t * *buffer*)

Get the total length of a scrambled object. TODO: Create a scramble_export() function, and/or alter the import interfaces to make using managed strings easier. ie. the import/export functions could provide wrappers around ddecrypt/encrypt which return strings instead of scramble objects. HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the total length, in bytes, of the scrambled object.

Definition at line 20 of file scramble.c.

References scramble_body_length(), scramble_head_t, and scramble_vector_length().

5.273.1.47 void* scramble_vector_data (scramble_t * *buffer*)

Get a pointer to the start of the IV from a scrambled data header.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

a pointer to the start of the scrambled data header's associated IV block.

Definition at line 92 of file scramble.c.

References scramble_head_t.

Referenced by scramble_decrypt(), and scramble_encrypt().

5.273.1.48 uint64_t scramble_vector_length (scramble_t * *buffer*)

Get the length of a scrambled object's IV.

Parameters:

buffer a pointer to the header of the scrambled data.

Returns:

the length, in bytes, of the scrambled object's IV.

Definition at line 70 of file scramble.c.

References scramble_head_t.

Referenced by scramble_body_data(), scramble_decrypt(), and scramble_total_length().

5.273.1.49 stringer_t* symmetric_decrypt (cipher_t * *cipher*, stringer_t * *vector*, stringer_t * *key*, stringer_t * *input*)

symmetric.c

Definition at line 129 of file symmetric.c.

References EVP_CIPHER_CTX_block_size_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_iv_length_d, EVP_CIPHER_CTX_key_length_d, EVP_CIPHER_flags_d, EVP_DecryptFinal_ex_d, EVP_DecryptInit_ex_d, EVP_DecryptUpdate_d, log_pedantic, MEMORYBUF, ssl_error_string(), st_alloc(), st_data_get(), st_empty_out(), st_free(), and st_length_set().

Referenced by scramble_decrypt().

5.273.1.50 stringer_t* symmetric_encrypt (cipher_t * *cipher*, stringer_t * *vector*, stringer_t * *key*, stringer_t * *input*)

HIGH: Alter the interfaces here to allow passing in the output buffer, or options which allow us to store the decrypted data in secure memory.
Encrypt a block of data using a symmetric cipher.

Parameters:

cipher the cipher type being used to encrypt the data.

vector the IV used for the encryption operation.

key the symmetric encryption key used to encrypt the data.

input a managed string containing the data to be encrypted.

Returns:

NULL on failure, or a pointer to a managed string containing the encrypted data.

Definition at line 21 of file symmetric.c.

References EVP_CIPHER_CTX_block_size_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_iv_length_d, EVP_CIPHER_CTX_key_length_d, EVP_CIPHER_flags_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, log_pedantic, st_alloc(), st_data_get(), st_empty_out(), st_free(), and st_length_set().

Referenced by scramble_encrypt().

5.273.1.51 stringer_t* symmetric_key (cipher_t * *cipher*, stringer_t * *key*, stringer_t * *output*)

Derive a symmetric key from a user's password.

Note:

Depending on the length of the password, either SHA1, SHA224, or SHA256 may be used as the hashing algorithm. If output is passed as NULL, a new managed string will be allocated to hold the output of the operation.

Parameters:

cipher the specified cipher type.

key the user's password, as a managed string.

output the output managed string to receive the digested password. Can be NULL.

Returns:

NULL on failure, or a pointer to the managed string containing the key.

Definition at line 289 of file symmetric.c.

References cipher_key_length(), hash_sha1(), hash_sha224(), hash_sha256(), st_length_get(), st_length_set(), st_output(), and st_valid_tracked().

Referenced by scramble_decrypt(), and scramble_encrypt().

5.273.1.52 stringer_t* symmetric_vector (cipher_t * cipher, stringer_t * output)

Generate an IV data suitable for the specified cipher.

Note:

If output is NULL, a new managed string will be allocated and returned by the function.

Parameters:

cipher the cipher type to be used.

output the managed string that will store the IV data.

Returns:

NULL on failure, or a pointer to the managed string that contains the IV data.

Definition at line 252 of file symmetric.c.

References cipher_vector_length(), log_pedantic, PLACER, rand_write(), st_data_get(), st_free(), st_length_set(), st_output(), and st_valid_tracked().

Referenced by scramble_encrypt().

5.274 src/providers/dime/common/crypto_pub.c File Reference

```
#include "dime/common/dcrypto.h"
#include "dime/common/error.h"
```

Functions

- int [crypto_init](#) (void)
- void [crypto_shutdown](#) (void)
- int [verify_ec_signature](#) (const unsigned char *hash, size_t hlen, const unsigned char *sig, size_t slen, EC_KEY *key)
- int [verify_ec_sha_signature](#) (const unsigned char *data, size_t dlen, unsigned int shabits, const unsigned char *sig, size_t slen, EC_KEY *key)
- unsigned char * [ec_sign_data](#) (const unsigned char *hash, size_t hlen, EC_KEY *key, size_t *siglen)
- unsigned char * [ec_sign_sha_data](#) (const unsigned char *data, size_t dlen, unsigned int shabits, EC_KEY *key, size_t *siglen)
- unsigned char * [serialize_ec_pubkey](#) (EC_KEY *key, size_t *outsize)
- EC_KEY * [deserialize_ec_pubkey](#) (const unsigned char *buf, size_t blen)
- unsigned char * [serialize_ec_privkey](#) (EC_KEY *key, size_t *outsize)
- EC_KEY * [deserialize_ec_privkey](#) (const unsigned char *buf, size_t blen)
- EC_KEY * [load_ec_privkey](#) (const char *filename)
- EC_KEY * [load_ec_pubkey](#) (const char *filename)
- EC_KEY * [generate_ec_keypair](#) (void)
- void [free_ec_key](#) (EC_KEY *key)
- ED25519_KEY * [generate_ed25519_keypair](#) (void)
- int [ed25519_sign_data](#) (const unsigned char *data, size_t dlen, ED25519_KEY *key, [ed25519_signature](#) sigbuf)
- int [ed25519_verify_sig](#) (const unsigned char *data, size_t dlen, ED25519_KEY *key, [ed25519_signature](#) sigbuf)
- void [free_ed25519_key](#) (ED25519_KEY *key)
- void [free_ed25519_key_chain](#) (ED25519_KEY **keys)
- ED25519_KEY * [load_ed25519_privkey](#) (const char *filename)
- ED25519_KEY * [deserialize_ed25519_pubkey](#) (const unsigned char *serial_pubkey)
- ED25519_KEY * [deserialize_ed25519_privkey](#) (const unsigned char *serial_privkey)
- void * [ecies_env_derivation](#) (const void *input, size_t ilen, void *output, size_t *olen)
- int [compute_aes256_kek](#) (EC_KEY *public_key, EC_KEY *private_key, unsigned char *keybuf)
- int [get_random_bytes](#) (void *buf, size_t len)
- int [encrypt_aes_256](#) (unsigned char *outbuf, const unsigned char *data, size_t dlen, const unsigned char *key, const unsigned char *iv)
- int [decrypt_aes_256](#) (unsigned char *outbuf, const unsigned char *data, size_t dlen, const unsigned char *key, const unsigned char *iv)

5.274.1 Function Documentation

5.274.1.1 int compute_aes256_kek (EC_KEY *public_key, EC_KEY *private_key, unsigned char *keybuf)

Definition at line 96 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.2 int crypto_init (void)

Definition at line 4 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.3 void crypto_shutdown (void)

Definition at line 8 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.274.1.4 int decrypt_aes_256 (unsigned char * *outbuf*, const unsigned char * *data*, size_t *dlen*, const unsigned char * *key*, const unsigned char * *iv*)

Definition at line 108 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.5 EC_KEY* deserialize_ec_privkey (const unsigned char * *buf*, size_t *blen*)

Definition at line 40 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.6 EC_KEY* deserialize_ec_pubkey (const unsigned char * *buf*, size_t *blen*)

Definition at line 32 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.7 ED25519_KEY* deserialize_ed25519_privkey (const unsigned char * *serial_privkey*)

Definition at line 88 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.8 ED25519_KEY* deserialize_ed25519_pubkey (const unsigned char * *serial_pubkey*)

Definition at line 84 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.9 unsigned char* ec_sign_data (const unsigned char * *hash*, size_t *hlen*, EC_KEY * *key*, size_t * *siglen*)

Definition at line 20 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.10 unsigned char* ec_sign_sha_data (const unsigned char * *data*, size_t *dlen*, unsigned int *shabits*, EC_KEY * *key*, size_t * *siglen*)

Definition at line 24 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.11 void* ecies_env_derivation (const void * *input*, size_t *ilen*, void * *output*, size_t * *olen*)

Definition at line 92 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.12 `int ed25519_sign_data (const unsigned char * data, size_t dlen, ED25519_KEY * key, ed25519_signature sigbuf)`

Definition at line 64 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL`.

5.274.1.13 `int ed25519_verify_sig (const unsigned char * data, size_t dlen, ED25519_KEY * key, ed25519_signature sigbuf)`

Definition at line 68 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL`.

5.274.1.14 `int encrypt_aes_256 (unsigned char * outbuf, const unsigned char * data, size_t dlen, const unsigned char * key, const unsigned char * iv)`

Definition at line 104 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL`.

5.274.1.15 `void free_ec_key (EC_KEY * key)`

Definition at line 56 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL_VOID`.

5.274.1.16 `void free_ed25519_key (ED25519_KEY * key)`

Definition at line 72 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL_VOID`.

5.274.1.17 `void free_ed25519_key_chain (ED25519_KEY ** keys)`

Definition at line 76 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL_VOID`.

5.274.1.18 `EC_KEY* generate_ec_keypair (void)`

Definition at line 52 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL`.

5.274.1.19 `ED25519_KEY* generate_ed25519_keypair (void)`

Definition at line 60 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL`.

5.274.1.20 `int get_random_bytes (void * buf, size_t len)`

Definition at line 100 of file `crypto_pub.c`.

References `PUBLIC_FUNC_IMPL`.

5.274.1.21 EC_KEY* load_ec_privkey (const char * *filename*)

Definition at line 44 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.22 EC_KEY* load_ec_pubkey (const char * *filename*)

Definition at line 48 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.23 ED25519_KEY* load_ed25519_privkey (const char * *filename*)

Definition at line 80 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.24 unsigned char* serialize_ec_privkey (EC_KEY * *key*, size_t * *outsize*)

Definition at line 36 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.25 unsigned char* serialize_ec_pubkey (EC_KEY * *key*, size_t * *outsize*)

Definition at line 28 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.26 int verify_ec_sha_signature (const unsigned char * *data*, size_t *dlen*, unsigned int *shabits*, const unsigned char * *sig*, size_t *slen*, EC_KEY * *key*)

Definition at line 16 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.274.1.27 int verify_ec_signature (const unsigned char * *hash*, size_t *hlen*, const unsigned char * *sig*, size_t *slen*, EC_KEY * *key*)

Definition at line 12 of file crypto_pub.c.

References PUBLIC_FUNC_IMPL.

5.275 src/providers/dime/common/dcrypto.h File Reference

```
#include <openssl/obj_mac.h>
#include <openssl/ec.h>
#include <openssl/ecdsa.h>
#include <openssl/pem.h>
#include "dime/ed25519/ed25519.h"
#include "dime/common/error.h"
```

Data Structures

- struct [ED25519_KEY](#)

Defines

- #define [AES_256_PADDING_SIZE](#) 16
- #define [AES_256_KEY_SIZE](#) 32
- #define [AES_256_KEY_SIZE](#) 48
- #define [EC_SIGNING_CURVE](#) NID_secp256k1
- #define [EC_ENCRYPT_CURVE](#) NID_secp256k1
- #define [ED25519_KEY_SIZE](#) 32
- #define [ED25519_KEY_B64_SIZE](#) 43
- #define [ED25519_SIG_SIZE](#) 64
- #define [ED25519_SIG_B64_SIZE](#) 86
- #define [EC_PUBKEY_SIZE](#) 33

Functions

- [PUBLIC_FUNC_DECL](#) (int, crypto_init, void)
- [PUBLIC_FUNC_DECL](#) (void, crypto_shutdown, void)
- [PUBLIC_FUNC_DECL](#) (EC_KEY *, load_ec_privkey, const char *filename)
- [PUBLIC_FUNC_DECL](#) (EC_KEY *, load_ec_pubkey, const char *filename)
- [PUBLIC_FUNC_DECL](#) (EC_KEY *, generate_ec_keypair, void)
- [PUBLIC_FUNC_DECL](#) (void, free_ec_key, EC_KEY *key)
- [PUBLIC_FUNC_DECL](#) (unsigned char *, ec_sign_data, const unsigned char *hash, size_t hlen, EC_KEY *key, size_t *siglen)
- [PUBLIC_FUNC_DECL](#) (unsigned char *, ec_sign_sha_data, const unsigned char *data, size_t dlen, unsigned int shabits, EC_KEY *key, size_t *siglen)
- [PUBLIC_FUNC_DECL](#) (int, verify_ec_signature, const unsigned char *hash, size_t hlen, const unsigned char *sig, size_t slen, EC_KEY *key)
- [PUBLIC_FUNC_DECL](#) (int, verify_ec_sha_signature, const unsigned char *data, size_t dlen, unsigned int shabits, const unsigned char *sig, size_t slen, EC_KEY *key)
- [PUBLIC_FUNC_DECL](#) (void *, ecies_env_derivation, const void *input, size_t ilen, void *output, size_t *olen)
- [PUBLIC_FUNC_DECL](#) (int, compute_aes256_kek, EC_KEY *public_key, EC_KEY *private_key, unsigned char *keybuf)
- [PUBLIC_FUNC_DECL](#) (unsigned char *, serialize_ec_pubkey, EC_KEY *key, size_t *outsize)
- [PUBLIC_FUNC_DECL](#) (EC_KEY *, deserialize_ec_pubkey, const unsigned char *buf, size_t blen)
- [PUBLIC_FUNC_DECL](#) (unsigned char *, serialize_ec_privkey, EC_KEY *key, size_t *outsize)
- [PUBLIC_FUNC_DECL](#) (EC_KEY *, deserialize_ec_privkey, const unsigned char *buf, size_t blen)
- [PUBLIC_FUNC_DECL](#) (ED25519_KEY *, load_ed25519_privkey, const char *filename)
- [PUBLIC_FUNC_DECL](#) (ED25519_KEY *, generate_ed25519_keypair, void)

- [PUBLIC_FUNC_DECL](#) (int, ed25519_sign_data, const unsigned char *data, size_t dlen, [ED25519_KEY](#) *key, ed25519_signature sigbuf)
- [PUBLIC_FUNC_DECL](#) (int, ed25519_verify_sig, const unsigned char *data, size_t dlen, [ED25519_KEY](#) *key, ed25519_signature sigbuf)
- [PUBLIC_FUNC_DECL](#) (void, free_ed25519_key, [ED25519_KEY](#) *key)
- [PUBLIC_FUNC_DECL](#) (void, free_ed25519_key_chain, [ED25519_KEY](#) **keys)
- [PUBLIC_FUNC_DECL](#) ([ED25519_KEY](#) *, deserialize_ed25519_pubkey, const unsigned char *serial_pubkey)
- [PUBLIC_FUNC_DECL](#) ([ED25519_KEY](#) *, deserialize_ed25519_privkey, const unsigned char *serial_privkey)
- [PUBLIC_FUNC_DECL](#) (int, encrypt_aes_256, unsigned char *outbuf, const unsigned char *data, size_t dlen, const unsigned char *key, const unsigned char *iv)
- [PUBLIC_FUNC_DECL](#) (int, decrypt_aes_256, unsigned char *outbuf, const unsigned char *data, size_t dlen, const unsigned char *key, const unsigned char *iv)
- [PUBLIC_FUNC_DECL](#) (int, get_random_bytes, void *buf, size_t len)

5.275.1 Define Documentation

5.275.1.1 #define AES_256_KEK_SIZE 48

Definition at line 14 of file dcrypto.h.

5.275.1.2 #define AES_256_KEY_SIZE 32

Definition at line 13 of file dcrypto.h.

Referenced by `__attribute__()`.

5.275.1.3 #define AES_256_PADDING_SIZE 16

Definition at line 12 of file dcrypto.h.

Referenced by `_decrypt_aes_256()`, and `_encrypt_aes_256()`.

5.275.1.4 #define EC_ENCRYPT_CURVE NID_secp256k1

Definition at line 17 of file dcrypto.h.

Referenced by `_crypto_init()`.

5.275.1.5 #define EC_PUBKEY_SIZE 33

Definition at line 23 of file dcrypto.h.

5.275.1.6 #define EC_SIGNING_CURVE NID_secp256k1

Definition at line 16 of file dcrypto.h.

5.275.1.7 #define ED25519_KEY_B64_SIZE 43

Definition at line 20 of file dcrypto.h.

Referenced by `_parse_dime_record()`.

5.275.1.8 #define ED25519_KEY_SIZE 32

Definition at line 19 of file dcrypto.h.

Referenced by _deserialize_dime_record_cb(), _deserialize_ed25519_privkey(), _deserialize_ed25519_pubkey(), _dump_dime_record_cb(), _load_ed25519_privkey(), _parse_dime_record(), and _serialize_dime_record_cb().

5.275.1.9 #define ED25519_SIG_B64_SIZE 86

Definition at line 22 of file dcrypto.h.

Referenced by _parse_dime_record().

5.275.1.10 #define ED25519_SIG_SIZE 64

Definition at line 21 of file dcrypto.h.

Referenced by _deserialize_dime_record_cb(), _dump_dime_record_cb(), _parse_dime_record(), and _serialize_dime_record_cb().

5.275.2 Function Documentation

- 5.275.2.1 PUBLIC_FUNC_DECL (int, get_random_bytes, void * *buf*, size_t *len*)
- 5.275.2.2 PUBLIC_FUNC_DECL (int, decrypt_aes_256, unsigned char * *outbuf*, const unsigned char * *data*, size_t *dlen*, const unsigned char * *key*, const unsigned char * *iv*)
- 5.275.2.3 PUBLIC_FUNC_DECL (int, encrypt_aes_256, unsigned char * *outbuf*, const unsigned char * *data*, size_t *dlen*, const unsigned char * *key*, const unsigned char * *iv*)
- 5.275.2.4 PUBLIC_FUNC_DECL (ED25519_KEY *, deserialize_ed25519_privkey, const unsigned char * *serial_privkey*)
- 5.275.2.5 PUBLIC_FUNC_DECL (ED25519_KEY *, deserialize_ed25519_pubkey, const unsigned char * *serial_pubkey*)
- 5.275.2.6 PUBLIC_FUNC_DECL (void, free_ed25519_key_chain, ED25519_KEY ** *keys*)
- 5.275.2.7 PUBLIC_FUNC_DECL (void, free_ed25519_key, ED25519_KEY * *key*)
- 5.275.2.8 PUBLIC_FUNC_DECL (int, ed25519_verify_sig, const unsigned char * *data*, size_t *dlen*, ED25519_KEY * *key*, ed25519_signature *sigbuf*)
- 5.275.2.9 PUBLIC_FUNC_DECL (int, ed25519_sign_data, const unsigned char * *data*, size_t *dlen*, ED25519_KEY * *key*, ed25519_signature *sigbuf*)
- 5.275.2.10 PUBLIC_FUNC_DECL (ED25519_KEY *, generate_ed25519_keypair, void)
- 5.275.2.11 PUBLIC_FUNC_DECL (ED25519_KEY *, load_ed25519_privkey, const char * *filename*)
- 5.275.2.12 PUBLIC_FUNC_DECL (EC_KEY *, deserialize_ec_privkey, const unsigned char * *buf*, size_t *blen*)
- 5.275.2.13 PUBLIC_FUNC_DECL (unsigned char *, serialize_ec_privkey, EC_KEY * *key*, size_t * *outsize*)
- 5.275.2.14 PUBLIC_FUNC_DECL (EC_KEY *, deserialize_ec_pubkey, const unsigned char * *buf*, size_t *blen*)
- 5.275.2.15 PUBLIC_FUNC_DECL (unsigned char *, serialize_ec_pubkey, EC_KEY * *key*, size_t * *outsize*)
- 5.275.2.16 PUBLIC_FUNC_DECL (int, compute_aes256_kek, EC_KEY * *public_key*, EC_KEY * *private_key*, unsigned char * *keybuf*)
- 5.275.2.17 PUBLIC_FUNC_DECL (void *, ecies_env_derivation, const void * *input*, size_t *ilen*, void * *output*, size_t * *olen*)
- 5.275.2.18 PUBLIC_FUNC_DECL (int, verify_ec_sha_signature, const unsigned char * *data*, size_t *dlen*, unsigned int *shabits*, const unsigned char * *sig*, size_t *slen*, EC_KEY * *key*)
- 5.275.2.19 PUBLIC_FUNC_DECL (int, verify_ec_signature, const unsigned char * *hash*, size_t *hlen*, const unsigned char * *sig*, size_t *slen*, EC_KEY * *key*)
- 5.275.2.20 PUBLIC_FUNC_DECL (unsigned char *, ec_sign_sha_data, const unsigned char * *data*, size_t *dlen*, unsigned int *shabits*, EC_KEY * *key*, size_t * *siglen*)
- 5.275.2.21 PUBLIC_FUNC_DECL (unsigned char *, ec_sign_data, const unsigned char * *hash*, size_t *hlen*, EC_KEY * *key*, size_t * *siglen*)
- 5.275.2.22 PUBLIC_FUNC_DECL (void, free_ec_key, EC_KEY * *key*)
- 5.275.2.23 PUBLIC_FUNC_DECL (EC_KEY *, generate_ec_keypair, void)
- ~~5.275.2.24 PUBLIC_FUNC_DECL (EC_KEY *, load_ec_pubkey, const char * *filename*)~~
- 5.275.2.25 PUBLIC_FUNC_DECL (EC_KEY *, load_ec_privkey, const char * *filename*)

5.276 src/providers/dime/common/error.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <netdb.h>
#include <openssl/buffer.h>
#include <openssl/err.h>
#include <openssl/ssl.h>
#include "dime/common/error.h"
#include "dime/common/misc.h"
#include "providers/symbols.h"
```

Functions

- struct `__attribute__((packed))`
- void `dump_last_error` (void)
- void `dump_error_stack` (void)

Dump the entire contents of the error stack to the console.
- void `_dump_error` (const `errinfo_t` *error)

Dump an error stack entry to the console.
- const char * `get_error_string` (unsigned int errcode)

Get the error string corresponding to a particular error code.
- unsigned int `get_last_error_code` (void)

Get the value of the last error code on the error stack.
- const `errinfo_t` * `get_last_error` (void)

Get the last error object on the error stack.
- `errinfo_t` * `pop_last_error` (void)
- const `errinfo_t` * `get_first_error` (void)
- void `_clear_error_stack` (void)

Clear the calling thread's error stack.
- `errinfo_t` * `_push_error_stack` (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *auxmsg)

Push an error onto the error stack.
- `errinfo_t` * `_push_error_stack_fmt` (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *fmt,...)

Push an error onto the error stack with a format-string supplied aux message.
- `errinfo_t` * `_push_error_stack_syscall` (const char *filename, const char *funcname, int lineno, int xerrno, const char *errfunc)

Push an error onto the error stack generated by a syscall (or anything that modifies errno).

- [errinfo_t * _push_error_stack_openssl](#) (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno)
Push an openssl-generated error onto the error stack.
- [errinfo_t * _push_error_stack_resolver](#) (const char *filename, const char *funcname, int lineno, int xerrno, int herrno, const char *errfunc)
Push an error onto the error stack generated by a resolver library function.
- [errinfo_t * _create_new_error](#) (errinfo_t *errptr, const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *auxmsg)
Populate a new entry on the error stack.

Variables

- `__thread struct thread_err_stack _t_err_stack`

5.276.1 Function Documentation

5.276.1.1 struct __attribute__((packed)) [read]

Definition at line 16 of file error.c.

References ERR_STACK_SIZE.

5.276.1.2 void _clear_error_stack (void)

Clear the calling thread's error stack.

Definition at line 178 of file error.c.

References `_t_err_stack`, and ERR_STACK_SIZE.

Referenced by `_destroy_cache_entry()`, `_do_ocsp_validation()`, `_dump_cache_data()`, `_dump_dime_record_cb()`, `_dump_dnskey_record_cb()`, `_evict_if_stale()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, `_get_signet()`, `_get_txt_record()`, `_load_cache_contents()`, `_lookup_dnskey()`, `_lookup_ds()`, `_save_cache_contents()`, `_unlink_object()`, and `_verify_certificate_callback()`.

5.276.1.3 errinfo_t* _create_new_error (errinfo_t * errptr, const char * filename, const char * funcname, int lineno, unsigned int errcode, int xerrno, const char * auxmsg)

Populate a new entry on the error stack.

Definition at line 316 of file error.c.

Referenced by `_push_error_stack()`.

5.276.1.4 void _dump_error (const errinfo_t * error)

Dump an error stack entry to the console.

Parameters:

error a pointer to the error stack object to be displayed.

Definition at line 83 of file error.c.

References `get_error_string()`.

Referenced by `dump_error_stack()`, and `dump_last_error()`.

5.276.1.5 `errinfo_t* _push_error_stack (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *auxmsg)`

Push an error onto the error stack.

Definition at line 192 of file `error.c`.

References `_create_new_error()`, `_t_err_stack`, and `ERR_STACK_SIZE`.

Referenced by `_push_error_stack_fmt()`, `_push_error_stack_openssl()`, `_push_error_stack_resolver()`, and `_push_error_stack_syscall()`.

5.276.1.6 `errinfo_t* _push_error_stack_fmt (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *fmt, ...)`

Push an error onto the error stack with a format-string supplied aux message.

Definition at line 217 of file `error.c`.

References `_push_error_stack()`.

5.276.1.7 `errinfo_t* _push_error_stack_openssl (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno)`

Push an openssl-generated error onto the error stack.

Definition at line 260 of file `error.c`.

References `_push_error_stack()`, `BUF_strlcat_d`, `ERR_error_string_n_d`, `ERR_get_error_d`, `ERR_load_crypto_strings_d`, `ERR_peek_error_line_data_d`, and `SSL_load_error_strings_d`.

5.276.1.8 `errinfo_t* _push_error_stack_resolver (const char *filename, const char *funcname, int lineno, int xerrno, int herrno, const char *errfunc)`

Push an error onto the error stack generated by a resolver library function.

Definition at line 302 of file `error.c`.

References `_push_error_stack()`, and `ERR_RESOLVER`.

5.276.1.9 `errinfo_t* _push_error_stack_syscall (const char *filename, const char *funcname, int lineno, int xerrno, const char *errfunc)`

Push an error onto the error stack generated by a syscall (or anything that modifies `errno`).

Parameters:

filename the name of the source file in which the error occurred.

funcname the name of the function in which the error occurred.

lineno the line in the source file in which the error occurred.

xerrno the immediate value of `errno` which was set by the error-returning function.

errfunc the name of the library call or syscall function responsible for setting `errno`.

Definition at line 240 of file `error.c`.

References `_push_error_stack()`, and `ERR_SYSCALL`.

5.276.1.10 void dump_error_stack (void)

Dump the entire contents of the error stack to the console.

Definition at line 55 of file error.c.

References `_dump_error()`, and `_t_err_stack`.

Referenced by `_destroy_cache_entry()`, `_do_ocsp_validation()`, `_dump_cache_data()`, `_dump_dime_record_cb()`, `_evict_if_stale()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_signet()`, `_get_txt_record()`, `_load_cache_contents()`, `_lookup_dnskey()`, `_lookup_ds()`, `_save_cache_contents()`, and `_unlink_object()`.

5.276.1.11 void dump_last_error (void)

Definition at line 36 of file error.c.

References `_dump_error()`, and `_t_err_stack`.

5.276.1.12 const char* get_error_string (unsigned int *errcode*)

Get the error string corresponding to a particular error code.

Parameters:

errcode the value of the error code to be looked up.

Returns:

a null-terminated string containing a description of the specified error code, or NULL on failure.

Definition at line 107 of file error.c.

References `err_desc_t::errmsg`.

Referenced by `_dump_error()`.

5.276.1.13 const errinfo_t* get_first_error (void)

Definition at line 165 of file error.c.

References `_t_err_stack`.

5.276.1.14 const errinfo_t* get_last_error (void)

Get the last error object on the error stack.

Returns:

a pointer to the last error object on the error stack, or NULL if the error stack is empty.

Definition at line 139 of file error.c.

References `_t_err_stack`.

Referenced by `_destroy_cache_entry()`, `_do_ocsp_validation()`, `_dump_cache_data()`, `_evict_if_stale()`, `_get_dime_record()`, `_get_signet()`, and `_unlink_object()`.

5.276.1.15 unsigned int get_last_error_code (void)

Get the value of the last error code on the error stack.

Returns:

the numerical error code of the last error in the error stack, or 0 if the stack was empty.

Definition at line 125 of file error.c.

References `_t_err_stack`.

5.276.1.16 `errinfo_t*` `pop_last_error` (void)

Definition at line 145 of file error.c.

References `_t_err_stack`.

5.276.2 Variable Documentation**5.276.2.1 `__thread struct thread_err_stack _t_err_stack`**

Definition at line 22 of file error.c.

Referenced by `_clear_error_stack()`, `_push_error_stack()`, `dump_error_stack()`, `dump_last_error()`, `get_first_error()`, `get_last_error()`, `get_last_error_code()`, and `pop_last_error()`.

5.277 src/providers/dime/common/error.h File Reference

```
#include <stdio.h>
#include <errno.h>
```

Data Structures

- struct [err_desc_t](#)

Defines

- #define [RET_ERROR_INT](#)(errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); return -1; } while (0)
- #define [RET_ERROR_UINT](#)(errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); return 0; } while (0)
- #define [RET_ERROR_PTR](#)(errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); return NULL; } while (0)
- #define [RET_ERROR_CUST](#)(retval, errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); return retval; } while (0)
- #define [RET_ERROR_INT_FMT](#)(errorcode, fmt,...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); return -1; } while (0)
- #define [RET_ERROR_UINT_FMT](#)(errorcode, fmt,...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); return 0; } while (0)
- #define [RET_ERROR_PTR_FMT](#)(errorcode, fmt,...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); return NULL; } while (0)
- #define [RET_ERROR_CUST_FMT](#)(retval, errorcode, fmt,...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); return retval; } while (0)
- #define [PUSH_ERROR](#)(errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); } while (0)
- #define [PUSH_ERROR_FMT](#)(errorcode, fmt,...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); } while (0)
- #define [PUSH_ERROR_SYSCALL](#)(func) do { _push_error_stack_syscall(__FILE__, __func__, __LINE__, errno, func); } while (0)
- #define [PUSH_ERROR_OPENSSL](#)() do { _push_error_stack_openssl(__FILE__, __func__, __LINE__, ERR_OPENSSL, errno); } while (0)
- #define [PUSH_ERROR_RESOLVER](#)(func) do { _push_error_stack_resolver(__FILE__, __func__, __LINE__, errno, h_errno, func); } while (0)
- #define [PUBLIC_FUNC_PROLOGUE](#)() { _clear_error_stack(); }
- #define [PUBLIC_FUNC_IMPL](#)(funcname,...) PUBLIC_FUNC_PROLOGUE(); return (__ ## funcname(__VA_ARGS__))
- #define [PUBLIC_FUNC_IMPL_VOID](#)(funcname,...) PUBLIC_FUNC_PROLOGUE(); _ ## funcname(__VA_ARGS__)
- #define [PUBLIC_FUNC_IMPL_VA1](#)(funcname, p1) PUBLIC_FUNC_PROLOGUE(); { va_list ap; va_start(ap, p1); __ ## funcname(p1, ap); va_end(ap); return; }
- #define [PUBLIC_FUNC_IMPL_VA1_RET](#)(ret, funcname, p1) PUBLIC_FUNC_PROLOGUE(); { va_list ap; ret result; va_start(ap, p1); result = __ ## funcname(p1, ap); va_end(ap); return result; }
- #define [PUBLIC_FUNC_IMPL_VA2](#)(funcname, p1, p2) PUBLIC_FUNC_PROLOGUE(); { va_list ap; va_start(ap, p2); __ ## funcname(p1, p2, ap); va_end(ap); return; }
- #define [PUBLIC_FUNC_IMPL_VA2_RET](#)(ret, funcname, p1, p2) PUBLIC_FUNC_PROLOGUE(); { va_list ap; ret result; va_start(ap, p2); result = __ ## funcname(p1, p2, ap); va_end(ap); return result; }
- #define [PUBLIC_FUNCTION_IMPLEMENT](#)(funcname,...) PUBLIC_FUNC_PROLOGUE(); return (funcname(__VA_ARGS__))
- #define [PUBLIC_FUNCTION_IMPLEMENT_VOID](#)(funcname,...) PUBLIC_FUNC_PROLOGUE(); funcname(__VA_ARGS__)
- #define [PUBLIC_FUNC_DECL](#)(rettype, funcname,...)
- #define [PUBLIC_FUNC_DECL_VA](#)(rettype, funcname,...)
- #define [ERR_SYSCALL](#) 1U
- #define [ERR_OPENSSL](#) 2U

- #define [ERR_RESOLVER](#) 3U
- #define [ERR_UNSPEC](#) 4U
- #define [ERR_BAD_PARAM](#) 5U
- #define [ERR_NOMEM](#) 6U
- #define [ERR_PERM](#) 7U
- #define [ERR_STACK_SIZE](#) 8U
- #define [ERR_CORRUPTION](#) 9U

Typedefs

- typedef struct errinfo [errinfo_t](#)

Functions

- struct [__attribute__\(\(packed\)\) errinfo](#)
- const char * [get_error_string](#) (unsigned int errcode)
Get the error string corresponding to a particular error code.
- const [errinfo_t](#) * [get_last_error](#) (void)
Get the last error object on the error stack.
- unsigned int [get_last_error_code](#) (void)
Get the value of the last error code on the error stack.
- [errinfo_t](#) * [pop_last_error](#) (void)
- const [errinfo_t](#) * [get_first_error](#) (void)
- void [dump_last_error](#) (void)
- void [dump_error_stack](#) (void)
Dump the entire contents of the error stack to the console.
- void [_clear_error_stack](#) (void)
Clear the calling thread's error stack.
- [errinfo_t](#) * [_push_error_stack](#) (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *auxmsg)
Push an error onto the error stack.
- [errinfo_t](#) * [_push_error_stack_fmt](#) (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *fmt,...) [__attribute__\(\(format printf\)\)](#)
- [errinfo_t](#) [errinfo_t](#) * [_push_error_stack_syscall](#) (const char *filename, const char *funcname, int lineno, int xerrno, const char *errfunc)
Push an error onto the error stack generated by a syscall (or anything that modifies errno).
- [errinfo_t](#) * [_push_error_stack_openssl](#) (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno)
Push an openssl-generated error onto the error stack.
- [errinfo_t](#) * [_push_error_stack_resolver](#) (const char *filename, const char *funcname, int lineno, int xerrno, int herrno, const char *errfunc)
Push an error onto the error stack generated by a resolver library function.
- [errinfo_t](#) * [_create_new_error](#) ([errinfo_t](#) *errptr, const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *auxmsg)

Populate a new entry on the error stack.

- void `_dump_error` (const `errinfo_t` *error)

Dump an error stack entry to the console.

5.277.1 Define Documentation

5.277.1.1 #define ERR_BAD_PARAM 5U

Definition at line 51 of file error.h.

Referenced by `__str_printf()`, `_add_cached_object()`, `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_add_dnskey_entry()`, `_add_dnskey_entry_rsa()`, `_add_ds_entry()`, `_b64decode()`, `_b64decode_nopad()`, `_b64encode()`, `_b64encode_nopad()`, `_b64encode_w_line separators()`, `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_clone_cached_object()`, `_clone_dnskey_record_cb()`, `_compute_aes256_kek()`, `_compute_dnskey_sha_hash()`, `_compute_sha_hash()`, `_compute_sha_hash_multibuf()`, `_connect_host()`, `_connect_timeout()`, `_count_ptr_chain()`, `_decode_rsa_pubkey()`, `_decrypt_aes_256()`, `_deserialize_array()`, `_deserialize_array_cb()`, `_deserialize_data()`, `_deserialize_dime_record_cb()`, `_deserialize_dnskey_record_cb()`, `_deserialize_ds_record_cb()`, `_deserialize_ec_privkey()`, `_deserialize_ec_pubkey()`, `_deserialize_ed25519_privkey()`, `_deserialize_ed25519_pubkey()`, `_deserialize_ocsp_response_cb()`, `_deserialize_signet_cb()`, `_deserialize_str_array()`, `_deserialize_string()`, `_deserialize_vardata()`, `_do_ocsp_validation()`, `_do_x509_hostname_check()`, `_do_x509_validation()`, `_domain_wildcard_check()`, `_dx_connect_dual()`, `_dx_connect_standard()`, `_ec_sign_data()`, `_ec_sign_sha_data()`, `_ed25519_sign_data()`, `_ed25519_verify_sig()`, `_encode_rsa_pubkey()`, `_encrypt_aes_256()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_get_cache_ocsp_id()`, `_get_cert_subject_cn()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, `_get_keytag()`, `_get_mx_records()`, `_get_random_bytes()`, `_get_rsa_dnskey()`, `_get_signet()`, `_get_txt_record()`, `_get_x509_cert_sha_hash()`, `_hex_encode()`, `_is_buf_zeroed()`, `_is_object_expired()`, `_is_validated_key()`, `_load_dnskey_file()`, `_load_ec_privkey()`, `_load_ec_pubkey()`, `_load_ed25519_privkey()`, `_lookup_ds()`, `_mem_append()`, `_mem_append_canon()`, `_mem_append_serialized()`, `_mem_append_serialized_array()`, `_mem_append_serialized_array_cb()`, `_mem_append_serialized_str_array()`, `_mem_append_serialized_string()`, `_parse_dime_record()`, `_ptr_chain_clone()`, `_read_file_data()`, `_read_pem_data()`, `_remove_cached_object()`, `_remove_cached_object_cmp()`, `_replace_object()`, `_rsa_verify_record()`, `_serialize_dime_record_cb()`, `_serialize_dnskey_record_cb()`, `_serialize_ds_record_cb()`, `_serialize_ec_privkey()`, `_serialize_ec_pubkey()`, `_serialize_ocsp_response_cb()`, `_serialize_signet_cb()`, `_set_cache_location()`, `_set_cache_permissions()`, `_sgnt_resolv_dmtcp_connect()`, `_sgnt_resolv_dmtcp_data()`, `_sgnt_resolv_dmtcp_ahlo()`, `_sgnt_resolv_dmtcp_expect_banner()`, `_sgnt_resolv_dmtcp_get_mode()`, `_sgnt_resolv_dmtcp_get_signet()`, `_sgnt_resolv_dmtcp_help()`, `_sgnt_resolv_dmtcp_history()`, `_sgnt_resolv_dmtcp_initiate_starttls()`, `_sgnt_resolv_dmtcp_issue_command()`, `_sgnt_resolv_dmtcp_mail_from()`, `_sgnt_resolv_dmtcp_noop()`, `_sgnt_resolv_dmtcp_quit()`, `_sgnt_resolv_dmtcp_rcpt_to()`, `_sgnt_resolv_dmtcp_reset()`, `_sgnt_resolv_dmtcp_send_and_read()`, `_sgnt_resolv_dmtcp_stats()`, `_sgnt_resolv_dmtcp_verify_signet()`, `_sgnt_resolv_dmtcp_write_data()`, `_sgnt_resolv_parse_line_code()`, `_sgnt_resolv_read_dmtcp_line()`, `_sgnt_resolv_read_dmtcp_multiline()`, `_sort_rrs_canonical()`, `_ssl_connect_host()`, `_unlink_object()`, `_validate_dime_record()`, `_validate_rrsig_rr()`, `_verify_dx_certificate()`, `_verify_ec_sha_signature()`, `_verify_ec_signature()`, `_write_pem_data()`, `dime_ctx_new()`, `dime_dmtcp_command_ahlo()`, `dime_dmtcp_command_helo()`, `dime_dmtcp_command_hist()`, `dime_dmtcp_command_mail()`, `dime_dmtcp_command_rcpt()`, `dime_dmtcp_command_sgnt_domain()`, `dime_dmtcp_command_sgnt_user()`, `dime_dmtcp_command_starttls()`, `dime_dmtcp_command_vrfy_domain()`, `dime_dmtcp_command_vrfy_user()`, `encrypt_ctx_new()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, `encrypt_keypair_generate()`, `libdime_base64_decode()`, and `libdime_base64_encode()`.

5.277.1.2 #define ERR_CORRUPTION 9U

Definition at line 57 of file error.h.

Referenced by `_read_pem_data()`.

5.277.1.3 #define ERR_NOMEM 6U

Definition at line 52 of file error.h.

Referenced by `__str_printf()`, `_add_cached_object()`, `_add_dnskey_entry_rsa()`, `_add_ds_entry()`, `_b64decode()`, `_b64decode_nopad()`, `_b64encode()`, `_b64encode_w_line separators()`, `_clone_cached_object()`, `_clone_dnskey_record_cb()`, `_compute_dnskey_sha_hash()`, `_create_cached_object()`, `_deserialize_array()`, `_deserialize_dime_record_cb()`, `_deserialize_dnskey_record_cb()`, `_deserialize_ds_record_cb()`, `_deserialize_ed25519_privkey()`, `_deserialize_ed25519_pubkey()`, `_deserialize_vardata()`, `_do_ocsp_validation()`, `_dx_connect_dual()`,

`_dx_connect_standard()`, `_encode_rsa_pubkey()`, `_generate_ed25519_keypair()`, `_get_cache_location()`, `_get_chr_date()`, `_get_dime_dir_location()`, `_get_dime_record()`, `_get_mx_records()`, `_get_signing_key_names()`, `_get_txt_record()`, `_hex_encode()`, `_load_cache_contents()`, `_load_ed25519_privkey()`, `_mem_append()`, `_mem_append_canon()`, `_mem_append_serialized()`, `_mem_append_serialized_array()`, `_mem_append_serialized_array_cb()`, `_mem_append_serialized_str_array()`, `_mem_append_serialized_string()`, `_parse_dime_record()`, `_ptr_chain_add()`, `_ptr_chain_clone()`, `_read_pem_data()`, `_rsa_verify_record()`, `_serialize_dime_record_cb()`, `_serialize_dnskey_record_cb()`, `_serialize_ds_record_cb()`, `_serialize_signet_cb()`, `_set_cache_location()`, `_sgnt_resolv_dmtpl_data()`, `_sgnt_resolv_dmtpl_ehlo()`, `_sgnt_resolv_dmtpl_get_signet()`, `_sgnt_resolv_dmtpl_mail_from()`, `_sgnt_resolv_dmtpl_rcpt_to()`, `_sgnt_resolv_read_dmtpl_line()`, `_sgnt_resolv_read_dmtpl_multiline()`, `dime_ctx_new()`, `encrypt_ctx_new()`, `libdime_base64_decode()`, and `libdime_base64_encode()`.

5.277.1.4 `#define ERR_OPENSSL 2U`

Definition at line 48 of file `error.h`.

5.277.1.5 `#define ERR_PERM 7U`

Definition at line 53 of file `error.h`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_load_cache_contents()`, `_remove_cached_object()`, `_remove_cached_object_cmp()`, and `_save_cache_contents()`.

5.277.1.6 `#define ERR_RESOLVER 3U`

Definition at line 49 of file `error.h`.

Referenced by `_push_error_stack_resolver()`.

5.277.1.7 `#define ERR_STACK_SIZE 8U`

Definition at line 56 of file `error.h`.

Referenced by `__attribute__()`, `_clear_error_stack()`, and `_push_error_stack()`.

5.277.1.8 `#define ERR_SYSCALL 1U`

Definition at line 47 of file `error.h`.

Referenced by `_push_error_stack_syscall()`, and `_write_pem_data()`.

5.277.1.9 `#define ERR_UNSPEC 4U`

Definition at line 50 of file `error.h`.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_add_dnskey_entry()`, `_add_dnskey_entry_rsa()`, `_add_ds_entry()`, `_b64decode()`, `_b64encode()`, `_b64encode_nopad()`, `_b64encode_w_line separators()`, `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_clone_cached_object()`, `_clone_dnskey_record_cb()`, `_compute_aes256_kek()`, `_compute_dnskey_sha_hash()`, `_compute_sha_hash()`, `_compute_sha_hash_multibuf()`, `_connect_host()`, `_connect_timeout()`, `_create_cached_object()`, `_crypto_init()`, `_decode_rsa_pubkey()`, `_decrypt_aes_256()`, `_deserialize_array()`, `_deserialize_array_cb()`, `_deserialize_data()`, `_deserialize_dime_record_cb()`, `_deserialize_dnskey_record_cb()`, `_deserialize_ds_record_cb()`, `_deserialize_ec_pubkey()`, `_deserialize_ocsp_response_cb()`, `_deserialize_signet_cb()`, `_deserialize_str_array()`, `_deserialize_string()`, `_deserialize_vardata()`, `_do_ocsp_validation()`, `_do_x509_validation()`, `_domain_wildcard_check()`, `_dx_connect_dual()`, `_dx_connect_standard()`, `_ec_sign_data()`, `_ec_sign_sha_data()`, `_encode_rsa_pubkey()`, `_encrypt_aes_256()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_generate_ed25519_keypair()`, `_get_cache_location()`, `_get_cache_obj_data()`, `_get_cache_ocsp_id()`, `_get_cert_store()`, `_get_cert_subject_cn()`, `_get_chr_date()`,

_get_dime_dir_location(), _get_dime_record(), _get_dime_record_from_file(), _get_ds_by_dnskey(), _get_mx_records(), _get_random_bytes(), _get_rsa_dnskey(), _get_signet(), _get_txt_record(), _get_x509_cert_sha_hash(), _initialize_resolver(), _is_object_expired(), _load_cache_contents(), _load_dnskey_file(), _load_ec_privkey(), _load_ec_pubkey(), _load_ed25519_privkey(), _lookup_dnskey(), _lookup_ds(), _mem_append_canon(), _parse_dime_record(), _ptr_chain_clone(), _read_file_data(), _read_pem_data(), _remove_cached_object(), _replace_object(), _rsa_verify_record(), _save_cache_contents(), _serialize_dnskey_record_cb(), _serialize_ec_privkey(), _serialize_ec_pubkey(), _serialize_ocsp_response_cb(), _serialize_signet_cb(), _sgnt_resolv_dmtcp_connect(), _sgnt_resolv_dmtcp_data(), _sgnt_resolv_dmtcp_ehlo(), _sgnt_resolv_dmtcp_expect_banner(), _sgnt_resolv_dmtcp_get_mode(), _sgnt_resolv_dmtcp_get_signet(), _sgnt_resolv_dmtcp_help(), _sgnt_resolv_dmtcp_history(), _sgnt_resolv_dmtcp_initiate_starttls(), _sgnt_resolv_dmtcp_issue_command(), _sgnt_resolv_dmtcp_mail_from(), _sgnt_resolv_dmtcp_noop(), _sgnt_resolv_dmtcp_quit(), _sgnt_resolv_dmtcp_rcpt_to(), _sgnt_resolv_dmtcp_reset(), _sgnt_resolv_dmtcp_send_and_read(), _sgnt_resolv_dmtcp_stats(), _sgnt_resolv_dmtcp_verify_signet(), _sgnt_resolv_dmtcp_write_data(), _sgnt_resolv_parse_line_code(), _sgnt_resolv_read_dmtcp_line(), _sort_rrs_canonical(), _ssl_connect_host(), _ssl_get_client_context(), _ssl_initialize(), _ssl_starttls(), _unlink_object(), _validate_dime_record(), _validate_rrsig_rr(), _validate_self_signed(), _verify_dx_certificate(), _verify_ec_sha_signature(), _verify_ec_signature(), dime_dmtcp_command_data(), dime_dmtcp_command_ehlo(), dime_dmtcp_command_helo(), dime_dmtcp_command_help(), dime_dmtcp_command_hist(), dime_dmtcp_command_mail(), dime_dmtcp_command_mode(), dime_dmtcp_command_noop(), dime_dmtcp_command_quit(), dime_dmtcp_command_rcpt(), dime_dmtcp_command_rset(), dime_dmtcp_command_sgnt_domain(), dime_dmtcp_command_sgnt_user(), dime_dmtcp_command_starttls(), dime_dmtcp_command_vrfy_domain(), and dime_dmtcp_command_vrfy_user().

5.277.1.10 #define PUBLIC_FUNC_DECL(rettype, funcname, ...)

Value:

```
rettype funcname(__VA_ARGS__); \
    rettype _ ## funcname(__VA_ARGS__)
```

Definition at line 38 of file error.h.

5.277.1.11 #define PUBLIC_FUNC_DECL_VA(rettype, funcname, ...)

Value:

```
rettype funcname(__VA_ARGS__, ...); \
    rettype _ ## funcname(__VA_ARGS__, ...); \
    rettype __ ## funcname(__VA_ARGS__, va_list ap)
```

Definition at line 42 of file error.h.

5.277.1.12 #define PUBLIC_FUNC_IMPL(funcname, ...) PUBLIC_FUNC_PROLOGUE(); return (_ ## funcname(__VA_ARGS__))

Definition at line 27 of file error.h.

Referenced by add_cached_object(), add_cached_object_cmp(), add_cached_object_cmp_forced(), add_cached_object_forced(), b64decode(), b64decode_nopad(), b64encode(), b64encode_nopad(), cached_object_exists(), cached_object_exists_cmp(), compute_aes256_kek(), compute_sha_hash(), compute_sha_hash_multibuf(), connect_host(), count_ptr_chain(), crypto_init(), decode_rsa_pubkey(), decrypt_aes_256(), deserialize_ec_privkey(), deserialize_ec_pubkey(), deserialize_ed25519_privkey(), deserialize_ed25519_pubkey(), do_ocsp_validation(), do_x509_hostname_check(), do_x509_validation(), dx_connect_dual(), dx_connect_standard(), ec_sign_data(), ec_sign_sha_data(), ecies_env_derivation(), ed25519_sign_data(), ed25519_verify_sig(), encode_rsa_pubkey(), encrypt_aes_256(), find_cached_object(), find_cached_object_cmp(), generate_ec_keypair(), generate_ed25519_keypair(), get_cache_location(), get_cache_obj_data(), get_cert_subject_cn(), get_chr_date(), get_dbg_level(), get_dime_dir_location(), get_dime_record(), get_dime_record_from_file(), get_random_bytes(), get_signet(), get_x509_cert_sha_hash(), hex_encode(), int_no_get_2b(), int_no_get_3b(), int_no_get_4b(), is_buf_zeroed(), load_cache_contents(), load_ec_privkey(), load_ec_pubkey(), load_ed25519_privkey(), mem_append(), parse_dime_record(), ptr_chain_add(), ptr_chain_clone(), read_file_data(), read_pem_data(), remove_cached_object(), remove_cached_object_cmp(), save_cache_contents(), serialize_ec_privkey(), serialize_ec_pubkey(), set_cache_location(), set_cache_permissions(), sgnt_resolv_dmtcp_connect(), sgnt_resolv_dmtcp_data(), sgnt_resolv_dmtcp_ehlo(), sgnt_resolv_dmtcp_get_mode(), sgnt_resolv_dmtcp_get_signet(), sgnt_resolv_dmtcp_help(), sgnt_resolv_dmtcp_history(), sgnt_resolv_dmtcp_mail_from(), sgnt_resolv_dmtcp_noop(), sgnt_resolv_dmtcp_quit(),

sgnt_resolv_dmtpt_rcpt_to(), sgnt_resolv_dmtpt_reset(), sgnt_resolv_dmtpt_stats(), sgnt_resolv_dmtpt_str_to_mode(), sgnt_resolv_dmtpt_verify_signet(), ssl_connect_host(), ssl_get_client_context(), ssl_initialize(), ssl_starttls(), validate_dime_record(), verify_dx_certificate(), verify_ec_sha_signature(), and verify_ec_signature().

5.277.1.13 `#define PUBLIC_FUNC_IMPL_VA1(funcname, p1) PUBLIC_FUNC_PROLOGUE(); { va_list ap; va_start(ap, p1); __
funcname(p1, ap); va_end(ap); return; }`

Definition at line 29 of file error.h.

5.277.1.14 `#define PUBLIC_FUNC_IMPL_VA1_RET(ret, funcname, p1) PUBLIC_FUNC_PROLOGUE(); { va_list ap; ret result;
va_start(ap, p1); result = __ ## funcname(p1, ap); va_end(ap); return result; }`

Definition at line 30 of file error.h.

5.277.1.15 `#define PUBLIC_FUNC_IMPL_VA2(funcname, p1, p2) PUBLIC_FUNC_PROLOGUE(); { va_list ap; va_start(ap,
p2); __ ## funcname(p1, p2, ap); va_end(ap); return; }`

Definition at line 31 of file error.h.

Referenced by dbgprint().

5.277.1.16 `#define PUBLIC_FUNC_IMPL_VA2_RET(ret, funcname, p1, p2) PUBLIC_FUNC_PROLOGUE(); { va_list ap; ret
result; va_start(ap, p2); result = __ ## funcname(p1, p2, ap); va_end(ap); return result; }`

Definition at line 32 of file error.h.

Referenced by str_printf().

5.277.1.17 `#define PUBLIC_FUNC_IMPL_VOID(funcname, ...) PUBLIC_FUNC_PROLOGUE(); _ ## funcname(__VA_ARGS__)`

Definition at line 28 of file error.h.

Referenced by crypto_shutdown(), destroy_cache_entry(), destroy_dime_record(), dump_buf(), dump_buf_outer(), free_ec_key(), free_ed25519_key(), free_ed25519_key_chain(), int_no_put_2b(), int_no_put_3b(), int_no_put_4b(), ptr_chain_free(), secure_wipe(), set_dbg_level(), sgnt_resolv_destroy_dmtpt_session(), ssl_disconnect(), and ssl_shutdown().

5.277.1.18 `#define PUBLIC_FUNC_PROLOGUE() { _clear_error_stack(); }`

Definition at line 24 of file error.h.

Referenced by dime_dmtpt_command_data(), dime_dmtpt_command_ehlo(), dime_dmtpt_command_helo(), dime_dmtpt_command_help(), dime_dmtpt_command_hist(), dime_dmtpt_command_mail(), dime_dmtpt_command_mode(), dime_dmtpt_command_noop(), dime_dmtpt_command_quit(), dime_dmtpt_command_rcpt(), dime_dmtpt_command_rset(), dime_dmtpt_command_sgnt_domain(), dime_dmtpt_command_sgnt_user(), dime_dmtpt_command_starttls(), dime_dmtpt_command_vrfy_domain(), and dime_dmtpt_command_vrfy_user().

5.277.1.19 `#define PUBLIC_FUNCTION_IMPLEMENT(funcname, ...) PUBLIC_FUNC_PROLOGUE(); return
(funcname(__VA_ARGS__))`

Definition at line 35 of file error.h.

Referenced by dime_dmsg_actor_to_string(), dime_dmsg_chunks_sig_origin_sign(), dime_dmsg_kek_in_derive(), dime_dmsg_message_binary_deserialize(), dime_dmsg_message_binary_serialize(), dime_dmsg_message_decrypt_as_auth(), dime_dmsg_message_decrypt_as_dest(), dime_dmsg_message_decrypt_as_orig(), dime_dmsg_message_decrypt_as_recp(), dime_dmsg_message_encrypt(), dime_dmsg_message_envelope_decrypt(), dime_dmsg_message_state_get(), dime_dmsg_object_chunk_create(), dime_dmsg_object_chunklist_destroy(), dime_dmsg_object_dump(), dime_dmsg_object_state_init(), dime_dmsg_object_state_to_string(), dime_dmtpt_command_

create(), dime_dmtplib_command_destroy(), dime_dmtplib_command_format(), dime_dmtplib_command_parse(), dime_keys_enckey_fetch(), dime_keys_file_create(), dime_keys_generate(), dime_keys_signkey_fetch(), dime_prsr_envelope_destroy(), dime_prsr_envelope_format(), dime_prsr_envelope_parse(), dime_prsr_headers_create(), dime_prsr_headers_format(), dime_prsr_headers_parse(), dime_sgnt_enckey_fetch(), dime_sgnt_enckey_set(), dime_sgnt_fid_count_get(), dime_sgnt_fid_exists(), dime_sgnt_fid_num_fetch(), dime_sgnt_fid_num_remove(), dime_sgnt_field_defined_create(), dime_sgnt_field_defined_set(), dime_sgnt_field_undefined_create(), dime_sgnt_field_undefined_fetch(), dime_sgnt_field_undefined_remove(), dime_sgnt_file_create(), dime_sgnt_fingerprint_crypto(), dime_sgnt_fingerprint_full(), dime_sgnt_fingerprint_id(), dime_sgnt_fingerprint_ssr(), dime_sgnt_id_fetch(), dime_sgnt_id_set(), dime_sgnt_msg_sig_verify(), dime_sgnt_sig_coc_sign(), dime_sgnt_sig_crypto_sign(), dime_sgnt_sig_full_sign(), dime_sgnt_sig_id_sign(), dime_sgnt_sig_ssr_sign(), dime_sgnt_signet_b64_deserialize(), dime_sgnt_signet_b64_serialize(), dime_sgnt_signet_binary_deserialize(), dime_sgnt_signet_binary_serialize(), dime_sgnt_signet_create(), dime_sgnt_signet_create_w_keys(), dime_sgnt_signet_crypto_split(), dime_sgnt_signet_dupe(), dime_sgnt_signet_full_split(), dime_sgnt_signet_load(), dime_sgnt_signkey_fetch(), dime_sgnt_signkey_set(), dime_sgnt_signkeys_msg_fetch(), dime_sgnt_signkeys_signet_fetch(), dime_sgnt_signkeys_software_fetch(), dime_sgnt_signkeys_tls_fetch(), dime_sgnt_sok_create(), dime_sgnt_sok_num_fetch(), dime_sgnt_state_to_str(), dime_sgnt_type_get(), dime_sgnt_type_set(), and dime_sgnt_validate_all().

5.277.1.20 **#define PUBLIC_FUNCTION_IMPLEMENT_VOID(funcname, ...) PUBLIC_FUNC_PROLOGUE(); funcname(__VA_ARGS__)**

Definition at line 36 of file error.h.

Referenced by dime_dmsg_message_destroy(), dime_dmsg_object_destroy(), dime_prsr_headers_destroy(), dime_sgnt_signet_destroy(), and dime_sgnt_signet_dump().

5.277.1.21 **#define PUSH_ERROR(errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); } while (0)**

Definition at line 17 of file error.h.

Referenced by _get_dime_record(), _sgnt_resolv_dmtplib_quit(), dime_dmtplib_command_data(), dime_dmtplib_command_ehlo(), dime_dmtplib_command_helo(), dime_dmtplib_command_help(), dime_dmtplib_command_hist(), dime_dmtplib_command_mail(), dime_dmtplib_command_mode(), dime_dmtplib_command_noop(), dime_dmtplib_command_quit(), dime_dmtplib_command_rcpt(), dime_dmtplib_command_rset(), dime_dmtplib_command_sgnt_domain(), dime_dmtplib_command_sgnt_user(), dime_dmtplib_command_starttls(), dime_dmtplib_command_vrfy_domain(), and dime_dmtplib_command_vrfy_user().

5.277.1.22 **#define PUSH_ERROR_FMT(errorcode, fmt, ...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); } while (0)**

Definition at line 18 of file error.h.

Referenced by _initialize_resolver(), _parse_dime_record(), _save_cache_contents(), _sgnt_resolv_dmtplib_data(), _sgnt_resolv_dmtplib_ehlo(), _sgnt_resolv_dmtplib_get_mode(), _sgnt_resolv_dmtplib_get_signet(), _sgnt_resolv_dmtplib_help(), _sgnt_resolv_dmtplib_history(), _sgnt_resolv_dmtplib_initiate_starttls(), _sgnt_resolv_dmtplib_mail_from(), _sgnt_resolv_dmtplib_noop(), _sgnt_resolv_dmtplib_quit(), _sgnt_resolv_dmtplib_rcpt_to(), _sgnt_resolv_dmtplib_reset(), _sgnt_resolv_dmtplib_stats(), and _sgnt_resolv_dmtplib_verify_signet().

5.277.1.23 **#define PUSH_ERROR_OPENSSL() do { _push_error_stack_openssl(__FILE__, __func__, __LINE__, ERR_OPENSSL, errno); } while (0)**

Definition at line 21 of file error.h.

Referenced by _compute_aes256_kek(), _compute_sha_hash(), _compute_sha_hash_multibuf(), _crypto_init(), _decode_rsa_pubkey(), _decrypt_aes_256(), _deserialize_ec_pubkey(), _deserialize_ocsp_response_cb(), _do_ocsp_validation(), _do_x509_validation(), _ec_sign_data(), _encode_rsa_pubkey(), _encrypt_aes_256(), _generate_ed25519_keypair(), _get_cache_ocsp_id(), _get_cert_store(), _get_cert_subject_cn(), _get_random_bytes(), _get_rsa_dnskey(), _get_x509_cert_sha_hash(), _rsa_verify_record(), _serialize_ec_privkey(), _serialize_ec_pubkey(), _serialize_ocsp_response_cb(), _sgnt_resolv_dmtplib_issue_command(), _sgnt_resolv_dmtplib_write_data(), _sgnt_resolv_read_dmtplib_line(), _ssl_connect_host(), _ssl_get_client_context(), _ssl_starttls(), _validate_self_signed(), _verify_dx_certificate(), and _verify_ec_signature().

5.277.1.24 `#define PUSH_ERROR_RESOLVER(func) do { _push_error_stack_resolver(__FILE__, __func__, __LINE__, errno, h_errno, func); } while (0)`

Definition at line 22 of file error.h.

Referenced by `_get_mx_records()`, `_get_txt_record()`, `_lookup_dnskey()`, `_lookup_ds()`, `_rsa_verify_record()`, `_sort_rrs_canonical()`, and `_validate_rrsig_rr()`.

5.277.1.25 `#define PUSH_ERROR_SYSCALL(func) do { _push_error_stack_syscall(__FILE__, __func__, __LINE__, errno, func); } while (0)`

Definition at line 20 of file error.h.

Referenced by `__str_printf()`, `_add_cached_object()`, `_add_dnskey_entry_rsa()`, `_add_ds_entry()`, `_b64decode()`, `_b64decode_nopad()`, `_b64encode()`, `_b64encode_w_line_separators()`, `_clone_cached_object()`, `_clone_dnskey_record_cb()`, `_connect_timeout()`, `_create_cached_object()`, `_deserialize_array()`, `_deserialize_dime_record_cb()`, `_deserialize_dnskey_record_cb()`, `_deserialize_ds_record_cb()`, `_deserialize_ed25519_privkey()`, `_deserialize_ed25519_pubkey()`, `_deserialize_string()`, `_deserialize_vardata()`, `_do_ocsp_validation()`, `_domain_wildcard_check()`, `_dx_connect_dual()`, `_dx_connect_standard()`, `_encode_rsa_pubkey()`, `_generate_ed25519_keypair()`, `_get_cache_location()`, `_get_cert_subject_cn()`, `_get_chr_date()`, `_get_dime_dir_location()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_mx_records()`, `_get_signing_key_names()`, `_get_txt_record()`, `_hex_encode()`, `_is_object_expired()`, `_load_cache_contents()`, `_load_dnskey_file()`, `_load_ed25519_privkey()`, `_mem_append()`, `_mem_append_canon()`, `_parse_dime_record()`, `_ptr_chain_add()`, `_ptr_chain_clone()`, `_read_file_data()`, `_read_pem_data()`, `_rsa_verify_record()`, `_save_cache_contents()`, `_serialize_signet_cb()`, `_set_cache_location()`, `_sgnt_resolv_dmtpl_data()`, `_sgnt_resolv_dmtpl_get_signet()`, `_sgnt_resolv_dmtpl_history()`, `_sgnt_resolv_dmtpl_issue_command()`, `_sgnt_resolv_dmtpl_verify_signet()`, `_sgnt_resolv_dmtpl_write_data()`, `_sgnt_resolv_read_dmtpl_line()`, `_sgnt_resolv_read_dmtpl_multiline()`, `_validate_rrsig_rr()`, and `_write_pem_data()`.

5.277.1.26 `#define RET_ERROR_CUST(retval, errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); return retval; } while (0)`

Definition at line 10 of file error.h.

Referenced by `_sgnt_resolv_dmtpl_get_mode()`, and `_sgnt_resolv_dmtpl_initiate_starttls()`.

5.277.1.27 `#define RET_ERROR_CUST_FMT(retval, errorcode, fmt, ...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); return retval; } while (0)`

Definition at line 15 of file error.h.

5.277.1.28 `#define RET_ERROR_INT(errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); return -1; } while (0)`

Definition at line 7 of file error.h.

Referenced by `__str_printf()`, `_cached_object_exists()`, `_cached_object_exists_cmp()`, `_compute_aes256_kek()`, `_compute_dnskey_sha_hash()`, `_compute_sha_hash()`, `_compute_sha_hash_multibuf()`, `_connect_host()`, `_connect_timeout()`, `_count_ptr_chain()`, `_crypto_init()`, `_decrypt_aes_256()`, `_do_ocsp_validation()`, `_do_x509_hostname_check()`, `_do_x509_validation()`, `_domain_wildcard_check()`, `_ed25519_sign_data()`, `_ed25519_verify_sig()`, `_encrypt_aes_256()`, `_get_random_bytes()`, `_get_x509_cert_sha_hash()`, `_initialize_resolver()`, `_is_buf_zeroed()`, `_is_object_expired()`, `_is_validated_key()`, `_load_cache_contents()`, `_load_dnskey_file()`, `_remove_cached_object()`, `_remove_cached_object_cmp()`, `_rsa_verify_record()`, `_save_cache_contents()`, `_set_cache_location()`, `_set_cache_permissions()`, `_sgnt_resolv_dmtpl_ehlo()`, `_sgnt_resolv_dmtpl_expect_banner()`, `_sgnt_resolv_dmtpl_issue_command()`, `_sgnt_resolv_dmtpl_mail_from()`, `_sgnt_resolv_dmtpl_noop()`, `_sgnt_resolv_dmtpl_quit()`, `_sgnt_resolv_dmtpl_rcpt_to()`, `_sgnt_resolv_dmtpl_reset()`, `_sgnt_resolv_dmtpl_verify_signet()`, `_sgnt_resolv_dmtpl_write_data()`, `_sort_rrs_canonical()`, `_ssl_initialize()`, `_validate_dime_record()`, `_validate_rrsig_rr()`, `_validate_self_signed()`, `_verify_dx_certificate()`, `_verify_ec_sha_signature()`, `_verify_ec_signature()`, and `_write_pem_data()`.

5.277.1.29 `#define RET_ERROR_INT_FMT(errorcode, fmt, ...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); return -1; } while (0)`

Definition at line 12 of file error.h.

Referenced by `_connect_host()`, `_do_ocsp_validation()`, `_do_x509_validation()`, `_load_dnskey_file()`, `_rsa_verify_record()`, `_validate_dime_record()`, `_validate_rrsig_rr()`, and `_write_pem_data()`.

5.277.1.30 `#define RET_ERROR_PTR(errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); return NULL; } while (0)`

Definition at line 9 of file error.h.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_add_dnskey_entry()`, `_add_dnskey_entry_rsa()`, `_add_ds_entry()`, `_b64decode()`, `_b64decode_nopad()`, `_b64encode()`, `_b64encode_nopad()`, `_b64encode_w_lineseparators()`, `_clone_cached_object()`, `_clone_dnskey_record_cb()`, `_create_cached_object()`, `_decode_rsa_pubkey()`, `_deserialize_dime_record_cb()`, `_deserialize_dnskey_record_cb()`, `_deserialize_ds_record_cb()`, `_deserialize_ec_privkey()`, `_deserialize_ec_pubkey()`, `_deserialize_ed25519_privkey()`, `_deserialize_ed25519_pubkey()`, `_deserialize_ocsp_response_cb()`, `_deserialize_signet_cb()`, `_deserialize_vardata()`, `_dx_connect_dual()`, `_dx_connect_standard()`, `_ec_sign_data()`, `_ec_sign_sha_data()`, `_encode_rsa_pubkey()`, `_find_cached_object()`, `_find_cached_object_cmp()`, `_generate_ed25519_keypair()`, `_get_cache_location()`, `_get_cache_obj_data()`, `_get_cache_ocsp_id()`, `_get_cert_store()`, `_get_cert_subject_cn()`, `_get_chr_date()`, `_get_dime_dir_location()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, `_get_mx_records()`, `_get_rsa_dnskey()`, `_get_signet()`, `_get_signing_key_names()`, `_get_txt_record()`, `_hex_encode()`, `_load_ec_privkey()`, `_load_ec_pubkey()`, `_load_ed25519_privkey()`, `_lookup_dnskey()`, `_lookup_ds()`, `_parse_dime_record()`, `_ptr_chain_add()`, `_ptr_chain_clone()`, `_read_file_data()`, `_read_pem_data()`, `_replace_object()`, `_serialize_dime_record_cb()`, `_serialize_dnskey_record_cb()`, `_serialize_ds_record_cb()`, `_serialize_ec_privkey()`, `_serialize_ec_pubkey()`, `_serialize_ocsp_response_cb()`, `_serialize_signet_cb()`, `_sgnt_resolv_dmtpt_connect()`, `_sgnt_resolv_dmtpt_data()`, `_sgnt_resolv_dmtpt_get_signet()`, `_sgnt_resolv_dmtpt_help()`, `_sgnt_resolv_dmtpt_history()`, `_sgnt_resolv_dmtpt_send_and_read()`, `_sgnt_resolv_dmtpt_stats()`, `_sgnt_resolv_parse_line_code()`, `_sgnt_resolv_read_dmtpt_line()`, `_sgnt_resolv_read_dmtpt_multiline()`, `_ssl_connect_host()`, `_ssl_get_client_context()`, `_ssl_starttls()`, and `_unlink_object()`.

5.277.1.31 `#define RET_ERROR_PTR_FMT(errorcode, fmt, ...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); return NULL; } while (0)`

Definition at line 14 of file error.h.

Referenced by `_add_cached_object()`, `_add_dnskey_entry()`, `_dx_connect_dual()`, `_get_ds_by_dnskey()`, `_get_mx_records()`, `_get_rsa_dnskey()`, `_get_txt_record()`, `_read_file_data()`, `_read_pem_data()`, and `_sgnt_resolv_parse_line_code()`.

5.277.1.32 `#define RET_ERROR_UINT(errorcode, auxmsg) do { _push_error_stack(__FILE__, __func__, __LINE__, errorcode, errno, auxmsg); return 0; } while (0)`

Definition at line 8 of file error.h.

Referenced by `_deserialize_array()`, `_deserialize_array_cb()`, `_deserialize_data()`, `_deserialize_str_array()`, `_deserialize_string()`, `_deserialize_vardata()`, `_get_keytag()`, `_mem_append()`, `_mem_append_canon()`, `_mem_append_serialized()`, `_mem_append_serialized_array()`, `_mem_append_serialized_array_cb()`, `_mem_append_serialized_str_array()`, and `_mem_append_serialized_string()`.

5.277.1.33 `#define RET_ERROR_UINT_FMT(errorcode, fmt, ...) do { _push_error_stack_fmt(__FILE__, __func__, __LINE__, errorcode, errno, fmt, __VA_ARGS__); return 0; } while (0)`

Definition at line 13 of file error.h.

Referenced by `_deserialize_data()`.

5.277.2 Typedef Documentation

5.277.2.1 typedef struct errinfo errinfo_t

Definition at line 74 of file error.h.

5.277.3 Function Documentation

5.277.3.1 struct __attribute__((packed)) [read]

Definition at line 65 of file error.h.

5.277.3.2 void _clear_error_stack (void)

Clear the calling thread's error stack.

Definition at line 178 of file error.c.

References `_t_err_stack`, and `ERR_STACK_SIZE`.

Referenced by `_destroy_cache_entry()`, `_do_ocsp_validation()`, `_dump_cache_data()`, `_dump_dime_record_cb()`, `_dump_dnskey_record_cb()`, `_evict_if_stale()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_dnskey_by_tag()`, `_get_ds_by_dnskey()`, `_get_signet()`, `_get_txt_record()`, `_load_cache_contents()`, `_lookup_dnskey()`, `_lookup_ds()`, `_save_cache_contents()`, `_unlink_object()`, and `_verify_certificate_callback()`.

5.277.3.3 errinfo_t* _create_new_error (errinfo_t * *errptr*, const char * *filename*, const char * *funcname*, int *lineno*, unsigned int *errcode*, int *xerrno*, const char * *auxmsg*)

Populate a new entry on the error stack.

Definition at line 316 of file error.c.

Referenced by `_push_error_stack()`.

5.277.3.4 void _dump_error (const errinfo_t * *error*)

Dump an error stack entry to the console.

Parameters:

error a pointer to the error stack object to be displayed.

Definition at line 83 of file error.c.

References `get_error_string()`.

Referenced by `dump_error_stack()`, and `dump_last_error()`.

5.277.3.5 errinfo_t* _push_error_stack (const char * *filename*, const char * *funcname*, int *lineno*, unsigned int *errcode*, int *xerrno*, const char * *auxmsg*)

Push an error onto the error stack.

Definition at line 192 of file error.c.

References `_create_new_error()`, `_t_err_stack`, and `ERR_STACK_SIZE`.

Referenced by `_push_error_stack_fmt()`, `_push_error_stack_openssl()`, `_push_error_stack_resolver()`, and `_push_error_stack_syscall()`.

5.277.3.6 `errinfo_t* _push_error_stack_fmt (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno, const char *fmt, ...)`

5.277.3.7 `errinfo_t* _push_error_stack_openssl (const char *filename, const char *funcname, int lineno, unsigned int errcode, int xerrno)`

Push an openssl-generated error onto the error stack.

Definition at line 260 of file error.c.

References `_push_error_stack()`, `BUF_strlcat_d`, `ERR_error_string_n_d`, `ERR_get_error_d`, `ERR_load_crypto_strings_d`, `ERR_peek_error_line_data_d`, and `SSL_load_error_strings_d`.

5.277.3.8 `errinfo_t* _push_error_stack_resolver (const char *filename, const char *funcname, int lineno, int xerrno, int herrno, const char *errfunc)`

Push an error onto the error stack generated by a resolver library function.

Definition at line 302 of file error.c.

References `_push_error_stack()`, and `ERR_RESOLVER`.

5.277.3.9 `errinfo_t errinfo_t* _push_error_stack_syscall (const char *filename, const char *funcname, int lineno, int xerrno, const char *errfunc)`

Push an error onto the error stack generated by a syscall (or anything that modifies errno).

Parameters:

filename the name of the source file in which the error occurred.

funcname the name of the function in which the error occurred.

lineno the line in the source file in which the error occurred.

xerrno the immediate value of errno which was set by the error-returning function.

errfunc the name of the library call or syscall function responsible for setting errno.

Definition at line 240 of file error.c.

References `_push_error_stack()`, and `ERR_SYSCALL`.

5.277.3.10 `void dump_error_stack (void)`

Dump the entire contents of the error stack to the console.

Definition at line 55 of file error.c.

References `_dump_error()`, and `_t_err_stack`.

Referenced by `_destroy_cache_entry()`, `_do_ocsp_validation()`, `_dump_cache_data()`, `_dump_dime_record_cb()`, `_evict_if_stale()`, `_get_dime_record()`, `_get_dime_record_from_file()`, `_get_signet()`, `_get_txt_record()`, `_load_cache_contents()`, `_lookup_dnskey()`, `_lookup_ds()`, `_save_cache_contents()`, and `_unlink_object()`.

5.277.3.11 `void dump_last_error (void)`

Definition at line 36 of file error.c.

References `_dump_error()`, and `_t_err_stack`.

5.277.3.12 `const char* get_error_string (unsigned int errcode)`

Get the error string corresponding to a particular error code.

Parameters:

errcode the value of the error code to be looked up.

Returns:

a null-terminated string containing a description of the specified error code, or NULL on failure.

Definition at line 107 of file error.c.

References `err_desc_t::errmsg`.

Referenced by `_dump_error()`.

5.277.3.13 `const errinfo_t* get_first_error (void)`

Definition at line 165 of file error.c.

References `_t_err_stack`.

5.277.3.14 `const errinfo_t* get_last_error (void)`

Get the last error object on the error stack.

Returns:

a pointer to the last error object on the error stack, or NULL if the error stack is empty.

Definition at line 139 of file error.c.

References `_t_err_stack`.

Referenced by `_destroy_cache_entry()`, `_do_ocsp_validation()`, `_dump_cache_data()`, `_evict_if_stale()`, `_get_dime_record()`, `_get_signet()`, and `_unlink_object()`.

5.277.3.15 `unsigned int get_last_error_code (void)`

Get the value of the last error code on the error stack.

Returns:

the numerical error code of the last error in the error stack, or 0 if the stack was empty.

Definition at line 125 of file error.c.

References `_t_err_stack`.

5.277.3.16 `errinfo_t* pop_last_error (void)`

Definition at line 145 of file error.c.

References `_t_err_stack`.

5.278 src/providers/dime/common/misc.c File Reference

```
#include <stdint.h>
#include <stdarg.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include <openssl/x509.h>
#include <openssl/err.h>
#include "dime/common/misc.h"
#include "dime/common/error.h"
#include "providers/symbols.h"
```

Functions

- `char * _b64encode_nopad` (unsigned char const *buf, size_t len)
Return the base64-encoded string of a data buffer without padding.
- `unsigned char * _b64decode_nopad` (char const *buf, size_t len, size_t *outlen)
Decode a base64-encoded string without any padding characters and return the result.
- `void _int_no_put_4b` (void *buf, uint32_t val)
Store a 4-byte value in a buffer in network byte order.
- `void _int_no_put_3b` (void *buf, uint32_t val)
Store a 3-byte value in a buffer in network byte order.
- `void _int_no_put_2b` (void *buf, uint16_t val)
Store a 2-byte value in a buffer in network byte order.
- `uint32_t _int_no_get_4b` (const void *buf)
Fetch a 4 byte value from a buffer and return it in host byte order.
- `uint32_t _int_no_get_3b` (const void *buf)
Fetch a 3 byte value from a buffer and return it in host byte order.
- `uint16_t _int_no_get_2b` (const void *buf)
Fetch a 2 byte value from a buffer and return it in host byte order.
- `char * _hex_encode` (unsigned char const *buf, size_t len)
Return a simple hex string representing a data buffer.
- `void _set_dbg_level` (unsigned int level)
Set the debugging level of the current process.
- `unsigned int _get_dbg_level` (void)

Get the debugging level of the current process.

- `int _str_printf (char **sbuf, const char *fmt,...)`
Append variable-argument formatted data to a dynamically allocated null-terminated string.
- `int __str_printf (char **sbuf, const char *fmt, va_list ap)`
Append variable-argument formatted data to a dynamically allocated null-terminated string.
- `size_t _mem_append (unsigned char **buf, size_t *blen, unsigned char const *data, size_t dlen)`
Append data to a dynamically allocated buffer.
- `void _ptr_chain_free (void *buf)`
Free a pointer chain and all its member elements.
- `void * _ptr_chain_add (void *buf, const void *addr)`
Append an address to the end of a pointer chain.
- `int _count_ptr_chain (void *buf)`
Count the number of elements in a pointer chain.
- `void * _ptr_chain_clone (void *buf)`
Clone a pointer chain.
- `char * _get_chr_date (time_t time, int local)`
Get the human-readable, US-specific representation of a specified UNIX time.
- `unsigned char * _b64decode (char const *buf, size_t len, size_t *outlen)`
Perform base64-decoding on a string and return the result.
- `char * _b64encode (unsigned char const *buf, size_t len)`
Return the base64-encoded string of a data buffer.
- `char * _b64encode_w_lineseparators (unsigned char const *buf, size_t len)`
- `void _dump_buf (unsigned char const *buf, size_t len, int all_hex)`
Dump a data buffer to the console for debugging purposes.
- `void _dump_buf_outer (unsigned char const *buf, size_t len, size_t router, int all_hex)`
Dump a data buffer to the console in abbreviated format for debugging purposes.
- `void _dbgprint (unsigned int dbglevel, const char *fmt,...)`
Print a debug string to the console.
- `void __dbgprint (unsigned int dbglevel, char const *fmt, va_list ap)`
Print a debug string to the console.
- `int _compute_sha_hash (size_t nbits, unsigned char const *buf, size_t blen, unsigned char *outbuf)`
Compute an SHA1, SHA256, or SHA512 hash of a block of data.
- `int _compute_sha_hash_multibuf (size_t nbits, sha_databuf_t *bufs, unsigned char *outbuf)`
Compute an SHA1, SHA256, or SHA512 hash of a block of data.
- `RSA * _decode_rsa_pubkey (unsigned char *data, size_t dlen)`
Decode an RSA public key from a buffer.

- unsigned char * [_encode_rsa_pubkey](#) (RSA *pubkey, size_t *enclen)
Encode and store an RSA public key into a buffer.
- int [_get_x509_cert_sha_hash](#) (X509 *cert, size_t nbits, unsigned char *out)
Get the SHA signature of an x509 certificate in DER format.
- int [_is_buf_zeroed](#) (void *buf, size_t len)
Check to see if a buffer is filled with null bytes.
- int [_compute_crc24_checksum](#) (void *buffer, size_t length)
Generate a 24-bit CRC value for the Privacy Enhanced Message format.
- int [_write_pem_data](#) (char const *b64_data, char const *checksum, char const *tag, char const *filename)
Create a pem file with specified tags and filename.
- unsigned char * [_read_file_data](#) (char const *filename, size_t *fsize)
Read the raw contents of a file into a buffer.
- char * [_read_pem_data](#) (char const *pemfile, char const *tag, int nospace)
Read the contents of a specified tag inside a PEM file into a buffer.
- void [_secure_wipe](#) (void *buf, size_t len)
Securely wipe a memory buffer, in preparation for deallocation.

Variables

- int [_verbose](#) = 0

5.278.1 Function Documentation

5.278.1.1 void [__dbgprint](#) (unsigned int *dbglevel*, char const **fmt*, va_list *ap*)

Print a debug string to the console.

Definition at line 957 of file misc.c.

References [_verbose](#).

Referenced by [_dbgprint\(\)](#).

5.278.1.2 int [__str_printf](#) (char ***sbuf*, const char **fmt*, va_list *ap*)

Append variable-argument formatted data to a dynamically allocated null-terminated string.

Definition at line 323 of file misc.c.

References [ERR_BAD_PARAM](#), [ERR_NOMEM](#), [PUSH_ERROR_SYSCALL](#), and [RET_ERROR_INT](#).

Referenced by [_str_printf\(\)](#).

5.278.1.3 unsigned char* _b64decode (char const * *buf*, size_t *len*, size_t * *outlen*)

Perform base64-decoding on a string and return the result.

Parameters:

- buf* a pointer to a buffer containing the data to be decoded.
- len* the length, in bytes, of the buffer to be decoded.
- outlen* a pointer to a variable to receive the length of the decoded data.

Returns:

- a pointer to a newly allocated buffer containing the base64-decoded, or NULL on failure. {free}

Definition at line 619 of file misc.c.

References B64_DECODED_LEN, ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

Referenced by _b64decode_nopad(), _load_dnskey_file(), _load_ec_privkey(), _load_ec_pubkey(), _load_ed25519_privkey(), and _read_pem_data().

5.278.1.4 unsigned char* _b64decode_nopad (char const * *buf*, size_t *len*, size_t * *outlen*)

Decode a base64-encoded string without any padding characters and return the result.

Parameters:

- buf* a pointer to a buffer containing the data to be decoded.
- len* the length, in bytes, of the buffer to be decoded.
- outlen* a pointer to a variable to receive the length of the decoded data.

Returns:

- a pointer to a newly allocated buffer containing the base64-decoded, or NULL on failure. {free}

Definition at line 78 of file misc.c.

References _b64decode(), ERR_BAD_PARAM, ERR_NOMEM, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

Referenced by _parse_dime_record().

5.278.1.5 char* _b64encode (unsigned char const * *buf*, size_t *len*)

Return the base64-encoded string of a data buffer.

Parameters:

- buf* a pointer to the data buffer to be base-64 encoded.
- len* the length, in bytes, of the buffer to be encoded.

Returns:

- a pointer to a newly allocated null-terminated string containing the base64-encoded data, or NULL on failure. {free}

Definition at line 705 of file misc.c.

References B64_ENCODED_LEN, ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

Referenced by _b64encode_nopad().

5.278.1.6 char* _b64encode_nopad (unsigned char const * *buf*, size_t *len*)

Return the base64-encoded string of a data buffer without padding.

Note:

This function ensures the smallest possible output length with no trailing '=' characters.

Parameters:

buf a pointer to the data buffer to be base-64 encoded.

len the length, in bytes, of the buffer to be encoded.

Returns:

a pointer to a newly allocated null-terminated string containing the base64-encoded data, or NULL on failure. {free}

Definition at line 42 of file misc.c.

References _b64encode(), ERR_BAD_PARAM, ERR_UNSPEC, and RET_ERROR_PTR.

Referenced by _dump_dime_record_cb().

5.278.1.7 char* _b64encode_w_lineseparators (unsigned char const * *buf*, size_t *len*)

Definition at line 769 of file misc.c.

References B64_ENCODED_LEN, ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

5.278.1.8 int _compute_crc24_checksum (void * *buffer*, size_t *length*)

Generate a 24-bit CRC value for the Privacy Enhanced Message format.

Note:

This function uses the predefined initial value and polynomial defined in RFC 4880.

Parameters:

buffer a pointer to the data to be checked.

length the length, in bytes, of the input buffer.

Returns:

the updated 24-bit CRC value in a 32-bit unsigned integer.

Definition at line 1350 of file misc.c.

Referenced by _read_pem_data().

5.278.1.9 int _compute_sha_hash (size_t *nbits*, unsigned char const * *buf*, size_t *blen*, unsigned char * *outbuf*)

Compute an SHA1, SHA256, or SHA512 hash of a block of data.

Parameters:

nbits specifies the number of bits for the hash operation (160, 256, and 512 are accepted values).

buf a pointer to a data buffer containing the data to be hashed.

blen the size, in bytes, of the buffer to be hashed.

outbuf a buffer that will contain the output of the hash operation (the caller is responsible for allocating the correct amount of space).

Returns:

0 on success or < 0 if an error was encountered.

Definition at line 991 of file misc.c.

References ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_OPENSSL, RET_ERROR_INT, SHA1_Final_d, SHA1_Init_d, SHA1_Update_d, SHA256_Final_d, SHA256_Init_d, SHA256_Update_d, SHA512_Final_d, SHA512_Init_d, and SHA512_Update_d.

Referenced by _add_cached_object(), _add_cached_object_cmp(), _compute_dnskey_sha_hash(), _ec_sign_sha_data(), _find_cached_object(), _get_cache_ocsp_id(), _get_x509_cert_sha_hash(), _remove_cached_object(), and _verify_ec_sha_signature().

5.278.1.10 int _compute_sha_hash_multibuf (size_t nbits, sha_databuf_t * bufs, unsigned char * outbuf)

Compute an SHA1, SHA256, or SHA512 hash of a block of data.

Parameters:

nbits specifies the number of bits for the hash operation (160, 256, and 512 are accepted values).

bufs a pointer to a data buffer containing the data to be hashed.

outbuf a buffer that will contain the output of the hash operation (the caller is responsible for allocating the correct amount of space).

Returns:

0 on success or < 0 if an error was encountered.

Definition at line 1048 of file misc.c.

References bufptr, data, ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_OPENSSL, RET_ERROR_INT, SHA1_Final_d, SHA1_Init_d, SHA1_Update_d, SHA256_Final_d, SHA256_Init_d, SHA256_Update_d, SHA512_Final_d, SHA512_Init_d, and SHA512_Update_d.

5.278.1.11 int _count_ptr_chain (void * buf)

Count the number of elements in a pointer chain.

Parameters:

buf the address of the pointer chain to be counted.

Returns:

the number of elements in the pointer chain, or -1 on general error.

Definition at line 505 of file misc.c.

References ERR_BAD_PARAM, and RET_ERROR_INT.

Referenced by _dump_dnskey_record_cb(), and _ptr_chain_clone().

5.278.1.12 void _dbgprint (unsigned int dbglevel, const char * fmt, ...)

Print a debug string to the console.

Parameters:

dbglevel the minimum necessary debugging level to be active in order for the data to be displayed.

fmt the format string specifier to be used to display the user data.

... the variable argument list specified by the format string.

Definition at line 945 of file misc.c.

References `__dbgprint()`.

Referenced by `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_add_dnskey_entry()`, `_connect_host()`, `_destroy_dime_record()`, `_do_ocsp_validation()`, `_do_x509_validation()`, `_dump_dns_header()`, `_dx_connect_dual()`, `_get_cache_location()`, `_get_dime_record()`, `_get_dnskey_by_tag()`, `_get_mx_records()`, `_get_rsa_dnskey()`, `_get_signet()`, `_get_txt_record()`, `_initialize_resolver()`, `_load_cache_contents()`, `_lookup_dnskey()`, `_lookup_ds()`, `_mem_append()`, `_parse_dime_record()`, `_rsa_verify_record()`, `_save_cache_contents()`, `_set_cache_location()`, `_sgnt_resolv_dmtplib_connect()`, `_sgnt_resolv_dmtplib_expect_banner()`, `_sgnt_resolv_dmtplib_issue_command()`, `_sgnt_resolv_dmtplib_write_data()`, `_sgnt_resolv_read_dmtplib_line()`, `_ssl_connect_host()`, `_ssl_disconnect()`, `_ssl_get_client_context()`, `_ssl_initialize()`, `_ssl_shutdown()`, `_ssl_starttls()`, `_unlink_object()`, `_validate_rrsig_rr()`, `_validate_self_signed()`, `_verify_certificate_callback()`, and `_verify_dx_certificate()`.

5.278.1.13 RSA* `_decode_rsa_pubkey` (unsigned char * *data*, size_t *dlen*)

Decode an RSA public key from a buffer.

Note:

This format is identical to the wire format of RSA public keys used for DNSSEC validation.

Parameters:

data a pointer to the data buffer that stores the encoded RSA public key in binary form.

dlen the length, in bytes, of the data buffer.

Returns:

a pointer to an RSA public key structure on success, or NULL on failure. {RSA_free}

Definition at line 1144 of file misc.c.

References `BN_bin2bn_d`, `BN_free_d`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, and `RSA_new_d`.

Referenced by `_deserialize_dnskey_record_cb()`.

5.278.1.14 void `_dump_buf` (unsigned char const * *buf*, size_t *len*, int *all_hex*)

Dump a data buffer to the console for debugging purposes.

Parameters:

buf a pointer to the data buffer to have its contents dumped.

len the size, in bytes, of the data buffer to be dumped.

all_hex if set, print all characters as hex codes - even printable ones.

Definition at line 849 of file misc.c.

Referenced by `_dump_buf_outer()`, `_dump_dnskey_record_cb()`, `_dump_ds_record_cb()`, `_get_txt_record()`, and `_mem_append()`.

5.278.1.15 void `_dump_buf_outer` (unsigned char const * *buf*, size_t *len*, size_t *nouter*, int *all_hex*)

Dump a data buffer to the console in abbreviated format for debugging purposes.

Note:

At most, only the first nouter leading and trailing bytes will be dumped.

Parameters:

buf a pointer to the data buffer to have its contents dumped.

len the size, in bytes, of the data buffer to be dumped.

nouter the number of outer (leading and trailing) bytes to be dumped.

all_hex if set, print all characters as hex codes - even printable ones.

Definition at line 893 of file misc.c.

References `_dump_buf()`.

Referenced by `_validate_rrsig_rr()`.

5.278.1.16 unsigned char* _encode_rsa_pubkey (RSA *pubkey, size_t *enclen)

Encode and store an RSA public key into a buffer.

Parameters:

pubkey a pointer to the RSA public key to be encoded.

enclen a pointer to a size_t that will receive the total size of the encoded public key on completion.

Returns:

a pointer to a buffer containing the encoded RSA public key as binary data on success, or NULL on failure. {free}

Definition at line 1223 of file misc.c.

References `BN_bn2bin_d`, `BN_num_bits_d`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_serialize_dnskey_record_cb()`.

5.278.1.17 char* _get_chr_date (time_t time, int local)

Get the human-readable, US-specific representation of a specified UNIX time.

Parameters:

time the time to be converted to a human-readable string.

local if set, return the time string as a local time; if not, as UTC.

Returns:

a string containing the formatted date-time; or NULL on errors. {free}

Definition at line 576 of file misc.c.

References `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_dump_cache()`, `_load_cache_contents()`, and `_validate_rrsig_rr()`.

5.278.1.18 unsigned int _get_dbg_level (void)

Get the debugging level of the current process.

Returns:

the current value of the debugging level.

Definition at line 298 of file misc.c.

References _verbose.

5.278.1.19 int _get_x509_cert_sha_hash (X509 * cert, size_t nbits, unsigned char * out)

Get the SHA signature of an x509 certificate in DER format.

Parameters:

cert a pointer to the x509 certificate to be hashed.

nbits the number of bits in the SHA operation (160, 256, and 512 are valid values).

out a pointer to a data buffer to receive the output (which the caller is responsible for allocating properly depending on the hash size).

Returns:

0 if the hash operation was completed successfully, or < 0 on error.

Definition at line 1290 of file misc.c.

References _compute_sha_hash(), CRYPTO_free_d, ERR_BAD_PARAM, ERR_UNSPEC, i2d_X509_d, PUSH_ERROR_OPENSSL, and RET_ERROR_INT.

Referenced by _verify_dx_certificate().

5.278.1.20 char* _hex_encode (unsigned char const * buf, size_t len)

Return a simple hex string representing a data buffer.

Parameters:

buf a pointer to the data buffer to be encoded as a hex string.

len the size, in bytes, of the buffer to be processed.

Returns:

NULL on failure, or a newly allocated null-terminated string containing the hex encoded input on success. {free}

Definition at line 253 of file misc.c.

References ERR_BAD_PARAM, ERR_NOMEM, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

Referenced by _dump_dime_record_cb().

5.278.1.21 uint16_t _int_no_get_2b (const void * buf)

Fetch a 2 byte value from a buffer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 2 bytes in the buffer in host byte order.

Definition at line 226 of file misc.c.

5.278.1.22 uint32_t _int_no_get_3b (const void * *buf*)

Fetch a 3 byte value from a buffer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 3 bytes in the buffer in host byte order.

Definition at line 203 of file misc.c.

5.278.1.23 uint32_t _int_no_get_4b (const void * *buf*)

Fetch a 4 byte value from a buffer and return it in host byte order.

Parameters:

buf a pointer to the data buffer from which the bytes will be read.

Returns:

the value of the first 4 bytes in the buffer in host byte order.

Definition at line 179 of file misc.c.

5.278.1.24 void _int_no_put_2b (void * *buf*, uint16_t *val*)

Store a 2-byte value in a buffer in network byte order.

Parameters:

buf a pointer to the data buffer where the value will be inserted.

val the 2 byte value to be placed in the specified buffer, in network byte order.

Definition at line 160 of file misc.c.

5.278.1.25 void _int_no_put_3b (void * *buf*, uint32_t *val*)

Store a 3-byte value in a buffer in network byte order.

Parameters:

buf a pointer to the data buffer where the value will be inserted.

val the 3 byte value to be placed in the specified buffer, in network byte order.

Definition at line 139 of file misc.c.

5.278.1.26 void _int_no_put_4b (void * *buf*, uint32_t *val*)

Store a 4-byte value in a buffer in network byte order.

Parameters:

buf a pointer to the data buffer where the value will be inserted.

val the 4 byte value to be placed in the specified buffer, in network byte order.

Definition at line 117 of file misc.c.

5.278.1.27 int _is_buf_zeroed (void * *buf*, size_t *len*)

Check to see if a buffer is filled with null bytes.

Parameters:

buf a pointer to the data buffer to be scanned.

len the size, in bytes, of the data buffer to be scanned.

Returns:

1 if the buffer was filled exclusively with null bytes, 0 if it was not, or -1 on general error.

Definition at line 1325 of file misc.c.

References ERR_BAD_PARAM, and RET_ERROR_INT.

5.278.1.28 size_t _mem_append (unsigned char ** *buf*, size_t * *blen*, unsigned char const * *data*, size_t *dlen*)

Append data to a dynamically allocated buffer.

Note:

This function should be called for the first time with *buf set to NULL.

Parameters:

buf a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.

blen a pointer to a variable that holds the buffer's current size and will receive the size of the newly resized output buffer.

data a pointer to a buffer holding the data that will be appended to the end of the output buffer.

dlen the size, in bytes, of the data buffer to be appended to the output buffer.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 385 of file misc.c.

References _dbgprint(), _dump_buf(), _verbose, ERR_BAD_PARAM, ERR_NOMEM, PUSH_ERROR_SYSCALL, and RET_ERROR_UINT.

Referenced by _compute_dnskey_sha_hash(), _get_mx_records(), _load_dnskey_file(), _mem_append_canon(), _mem_append_serialized(), _mem_append_serialized_array(), _mem_append_serialized_array_cb(), _mem_append_serialized_str_array(), _mem_append_serialized_string(), _rsa_verify_record(), _serialize_dime_record_cb(), _serialize_dnskey_record_cb(), and _serialize_ds_record_cb().

5.278.1.29 void* _ptr_chain_add (void * *buf*, const void * *addr*)

Append an address to the end of a pointer chain.

Parameters:

buf the address of the target pointer chain. If NULL, allocate a new one for the caller.

addr the address to be appended to the pointer chain.

Definition at line 456 of file misc.c.

References ERR_NOMEM, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

Referenced by _deserialize_array(), _deserialize_array_cb(), _deserialize_str_array(), _lookup_dnskey(), _lookup_ds(), and _parse_dime_record().

5.278.1.30 void* _ptr_chain_clone (void * *buf*)

Clone a pointer chain.

Parameters:

buf a pointer to the pointer chain to be cloned.

Returns:

a pointer to the cloned pointer chain on success, or NULL on failure. {ptr_chain_free}

Definition at line 531 of file misc.c.

References _count_ptr_chain(), ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

Referenced by _clone_dnskey_record_cb().

5.278.1.31 void _ptr_chain_free (void * *buf*)

Free a pointer chain and all its member elements.

Note:

All pointers in the chain will be free()'ed. To avoid this behavior, call free() on the pointer chain instead.

Parameters:

buf the address of the pointer chain to be freed.

Definition at line 431 of file misc.c.

Referenced by _deserialize_array(), _deserialize_array_cb(), _deserialize_str_array(), and _destroy_dime_record().

5.278.1.32 unsigned char* _read_file_data (char const * *filename*, size_t * *fsize*)

Read the raw contents of a file into a buffer.

Parameters:

filename a null-terminated string with the name of the file to have its contents read into memory.

fsize a pointer to a variable that will receive the length, in bytes, of the read data.

Returns:

a pointer to a buffer containing the raw contents of the specified file, or NULL on failure. {free}

Definition at line 1471 of file misc.c.

References ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, RET_ERROR_PTR, and RET_ERROR_PTR_FMT.

5.278.1.33 char* _read_pem_data (char const * *pemfile*, char const * *tag*, int *nospace*)

Read the contents of a specified tag inside a PEM file into a buffer.

Note:

This function will remove all newline characters inside the tag being read.

Parameters:

pemfile the name of the PEM file to be parsed.

tag the name of a tag in the file to have its contents read into memory.

nospace if set, strip all whitespace from the PEM file content.

Returns:

a pointer to a buffer containing the contents of the specified PEM file tag, or NULL on failure. {free}

Definition at line 1532 of file misc.c.

References _b64decode(), _compute_crc24_checksum(), _str_printf(), dbgprint(), ERR_BAD_PARAM, ERR_CORRUPTION, ERR_NOMEM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, RET_ERROR_PTR, and RET_ERROR_PTR_FMT.

Referenced by _load_ec_privkey(), _load_ec_pubkey(), and _load_ed25519_privkey().

5.278.1.34 void _secure_wipe (void * *buf*, size_t *len*)

Securely wipe a memory buffer, in preparation for deallocation.

Parameters:

buf a pointer to the data buffer to be wiped.

len the size, in bytes, of the data buffer to be wiped.

Definition at line 1719 of file misc.c.

Referenced by _compute_aes256_kek(), _free_ed25519_key(), _generate_ed25519_keypair(), _load_ec_privkey(), _load_ec_pubkey(), _load_ed25519_privkey(), and _write_pem_data().

5.278.1.35 void _set_dbg_level (unsigned int *level*)

Set the debugging level of the current process.

Note:

The debugging level determines which debug strings are displayed to the console. If a debug string is printed with a level equal to or higher than the debugging level, it will be displayed to the user; otherwise it will be suppressed.

Parameters:

level the value of the new debugging level to be set.

Definition at line 288 of file misc.c.

References _verbose.

5.278.1.36 `int _str_printf(char **sbuf, const char *fmt, ...)`

Append variable-argument formatted data to a dynamically allocated null-terminated string.

Definition at line 307 of file misc.c.

References `__str_printf()`.

Referenced by `_get_signing_key_names()`, `_read_pem_data()`, `_sgnt_resolv_dmtpl_ehlo()`, `_sgnt_resolv_dmtpl_mail_from()`, `_sgnt_resolv_dmtpl_rcpt_to()`, and `_sgnt_resolv_dmtpl_stats()`.

5.278.1.37 `int _write_pem_data(char const *b64_data, char const *checksum, char const *tag, char const *filename)`

Create a pem file with specified tags and filename.

Parameters:

b64_data Null terminated base64 encoded data.

tag Null terminated ASCII string containing the desired PEM tag.

filename Null terminated string containing the desired filename.

Returns:

0 on success, -1 on failure.

Definition at line 1379 of file misc.c.

References `_secure_wipe()`, `ERR_BAD_PARAM`, `ERR_SYSCALL`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, and `RET_ERROR_INT_FMT`.

5.278.2 Variable Documentation

5.278.2.1 `int _verbose = 0`

Definition at line 16 of file misc.c.

Referenced by `__dbgprint()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_do_ocsp_validation()`, `_do_x509_validation()`, `_get_dbg_level()`, `_get_txt_record()`, `_load_cache_contents()`, `_mem_append()`, `_ocsp_response_callback()`, `_set_dbg_level()`, `_unlink_object()`, and `_validate_rrsig_rr()`.

5.279 src/providers/dime/common/misc.h File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <ctype.h>
#include <time.h>
#include <openssl/ec.h>
#include <openssl/bio.h>
#include <openssl/evp.h>
#include <openssl/buffer.h>
#include <openssl/sha.h>
#include <openssl/rsa.h>
#include <openssl/bn.h>
#include "dime/common/error.h"
```

Data Structures

- struct [sha_databuf_t](#)

Defines

- #define [ANSI_COLOR_RED](#) "\x1b[31m"
- #define [ANSI_COLOR_RESET](#) "\x1b[0m"
- #define [ALERT_PRINT](#)(fp, x) { fprintf(fp, ANSI_COLOR_RED); x; fprintf(fp, ANSI_COLOR_RESET); fflush(fp); }
- #define [SHA_160_SIZE](#) 20
- #define [SHA_256_SIZE](#) 32
- #define [SHA_512_SIZE](#) 64
- #define [SHA_512_B64_SIZE](#) 86
- #define [B64_ENCODED_LEN](#)(len) (((len) + (((len) % 3) ? (3 - ((len) % 3)) : 0)) / 3) * 4)
- #define [B64_DECODED_LEN](#)(len) ((len) * 3 / 4)
- #define [chr_isspace](#)(c) isspace((unsigned char)(c))
- #define [chr_isprint](#)(c) isprint((unsigned char)(c))

Functions

- [PUBLIC_FUNC_DECL](#) (void, set_dbg_level, unsigned int level)
- [PUBLIC_FUNC_DECL](#) (unsigned int, get_dbg_level, void)
- [PUBLIC_FUNC_DECL](#) (uint32_t, int_no_get_4b, const void *buf)
- [PUBLIC_FUNC_DECL](#) (uint32_t, int_no_get_3b, const void *buf)
- [PUBLIC_FUNC_DECL](#) (uint16_t, int_no_get_2b, const void *buf)
- [PUBLIC_FUNC_DECL](#) (void, int_no_put_4b, void *buf, uint32_t val)
- [PUBLIC_FUNC_DECL](#) (void, int_no_put_3b, void *buf, uint32_t val)
- [PUBLIC_FUNC_DECL](#) (void, int_no_put_2b, void *buf, uint16_t val)
- [PUBLIC_FUNC_DECL_VA](#) (int, str_printf, char **sbuf, const char *fmt)
- [PUBLIC_FUNC_DECL](#) (size_t, mem_append, unsigned char **buf, size_t *blen, const unsigned char *data, size_t dlen)
- [PUBLIC_FUNC_DECL](#) (void *, ptr_chain_add, void *buf, const void *addr)

- [PUBLIC_FUNC_DECL](#) (void, ptr_chain_free, void *buf)
- [PUBLIC_FUNC_DECL](#) (int, count_ptr_chain, void *buf)
- [PUBLIC_FUNC_DECL](#) (void *, ptr_chain_clone, void *buf)
- [PUBLIC_FUNC_DECL](#) (char *, get_chr_date, time_t time, int local)
- [PUBLIC_FUNC_DECL](#) (int, is_buf_zeroed, void *buf, size_t len)
- [PUBLIC_FUNC_DECL](#) (void, secure_wipe, void *buf, size_t len)
- [PUBLIC_FUNC_DECL](#) (unsigned char *, b64decode, const char *buf, size_t len, size_t *outlen)
- [PUBLIC_FUNC_DECL](#) (unsigned char *, b64decode_nopad, const char *buf, size_t len, size_t *outlen)
- [PUBLIC_FUNC_DECL](#) (char *, b64encode, const unsigned char *buf, size_t len)
- [PUBLIC_FUNC_DECL](#) (char *, b64encode_nopad, const unsigned char *buf, size_t len)
- [PUBLIC_FUNC_DECL](#) (char *, b64encode_w_lineseparators, const unsigned char *buf, size_t len)
- [PUBLIC_FUNC_DECL](#) (char *, hex_encode, const unsigned char *buf, size_t len)
- [PUBLIC_FUNC_DECL](#) (void, dump_buf, const unsigned char *buf, size_t len, int all_hex)
- [PUBLIC_FUNC_DECL](#) (void, dump_buf_outer, const unsigned char *buf, size_t len, size_t nouter, int all_hex)
- [PUBLIC_FUNC_DECL_VA](#) (void, dbgprint, unsigned int dbglevel, const char *fmt)
- [PUBLIC_FUNC_DECL](#) (int, write_pem_data, const char *b64_data, const char *checksum, const char *tag, const char *filename)
- [PUBLIC_FUNC_DECL](#) (unsigned char *, read_file_data, const char *filename, size_t *fsize)
- [PUBLIC_FUNC_DECL](#) (char *, read_pem_data, const char *pemfile, const char *tag, int nospace)
- [PUBLIC_FUNC_DECL](#) (int, compute_sha_hash, size_t nbits, const unsigned char *buf, size_t blen, unsigned char *outbuf)
- [PUBLIC_FUNC_DECL](#) (int, compute_sha_hash_multibuf, size_t nbits, [sha_databuf_t](#) *bufs, unsigned char *outbuf)
- [PUBLIC_FUNC_DECL](#) (RSA *, decode_rsa_pubkey, unsigned char *data, size_t dlen)
- [PUBLIC_FUNC_DECL](#) (unsigned char *, encode_rsa_pubkey, RSA *pubkey, size_t *enclen)
- [PUBLIC_FUNC_DECL](#) (int, get_x509_cert_sha_hash, X509 *cert, size_t nbits, unsigned char *out)
- [PUBLIC_FUNC_DECL](#) (int, _compute_crc24_checksum, void *buffer, size_t length)

Variables

- [int _verbose](#)

5.279.1 Define Documentation

5.279.1.1 `#define ALERT_PRINT(fp, x) { fprintf(fp, ANSI_COLOR_RED); x; fprintf(fp, ANSI_COLOR_RESET); fflush(fp); }`

Definition at line 22 of file misc.h.

5.279.1.2 `#define ANSI_COLOR_RED "\x1b[31m"`

Definition at line 20 of file misc.h.

5.279.1.3 `#define ANSI_COLOR_RESET "\x1b[0m"`

Definition at line 21 of file misc.h.

5.279.1.4 `#define B64_DECODED_LEN(len) ((len) * 3 / 4)`

Definition at line 31 of file misc.h.

Referenced by [_b64decode\(\)](#).

5.279.1.5 #define B64_ENCODED_LEN(len) (((len) + ((len) % 3) ? (3 - ((len) % 3)) : 0) / 3) * 4)

Definition at line 30 of file misc.h.

Referenced by `_b64encode()`, and `_b64encode_w_line separators()`.

5.279.1.6 #define chr_isprint(c) isprint((unsigned char)(c))

Definition at line 34 of file misc.h.

Referenced by `_get_dime_record_from_file()`.

5.279.1.7 #define chr_isspace(c) isspace((unsigned char)(c))

Definition at line 33 of file misc.h.

Referenced by `_load_dnskey_file()`, `_parse_dime_record()`, `_sgnt_resolv_dmtplib_get_signt()`, `_sgnt_resolv_dmtplib_initiate_starttls()`, `_sgnt_resolv_dmtplib_stats()`, and `_sgnt_resolv_parse_line_code()`.

5.279.1.8 #define SHA_160_SIZE 20

Definition at line 24 of file misc.h.

Referenced by `_dump_dnskey_record_cb()`, `_get_cache_ocsp_id()`, and `_get_ds_by_dnskey()`.

5.279.1.9 #define SHA_256_SIZE 32

Definition at line 25 of file misc.h.

Referenced by `_add_cached_object()`, `_add_cached_object_cmp()`, `_cached_object_exists()`, `_dump_dnskey_record_cb()`, `_find_cached_object()`, `_get_ds_by_dnskey()`, `_remove_cached_object()`, and `_replace_object()`.

5.279.1.10 #define SHA_512_B64_SIZE 86

Definition at line 28 of file misc.h.

5.279.1.11 #define SHA_512_SIZE 64

Definition at line 26 of file misc.h.

Referenced by `_compute_aes256_kek()`, `_dump_dnskey_record_cb()`, `_ec_sign_sha_data()`, `_verify_dx_certificate()`, and `_verify_ec_sha_signature()`.

5.279.2 Function Documentation

- 5.279.2.1 PUBLIC_FUNC_DECL (int, *_compute_crc24_checksum*, void * *buffer*, size_t *length*)
- 5.279.2.2 PUBLIC_FUNC_DECL (int, *get_x509_cert_sha_hash*, X509 * *cert*, size_t *nbits*, unsigned char * *out*)
- 5.279.2.3 PUBLIC_FUNC_DECL (unsigned char *, *encode_rsa_pubkey*, RSA * *pubkey*, size_t * *enclen*)
- 5.279.2.4 PUBLIC_FUNC_DECL (RSA *, *decode_rsa_pubkey*, unsigned char * *data*, size_t *dlen*)
- 5.279.2.5 PUBLIC_FUNC_DECL (int, *compute_sha_hash_multibuf*, size_t *nbits*, sha_databuf_t * *bufs*, unsigned char * *outbuf*)
- 5.279.2.6 PUBLIC_FUNC_DECL (int, *compute_sha_hash*, size_t *nbits*, const unsigned char * *buf*, size_t *blen*, unsigned char * *outbuf*)
- 5.279.2.7 PUBLIC_FUNC_DECL (char *, *read_pem_data*, const char * *pemfile*, const char * *tag*, int *nospace*)
- 5.279.2.8 PUBLIC_FUNC_DECL (unsigned char *, *read_file_data*, const char * *filename*, size_t * *fsize*)
- 5.279.2.9 PUBLIC_FUNC_DECL (int, *write_pem_data*, const char * *b64_data*, const char * *checksum*, const char * *tag*, const char * *filename*)
- 5.279.2.10 PUBLIC_FUNC_DECL (void, *dump_buf_outer*, const unsigned char * *buf*, size_t *len*, size_t *nouter*, int *all_hex*)
- 5.279.2.11 PUBLIC_FUNC_DECL (void, *dump_buf*, const unsigned char * *buf*, size_t *len*, int *all_hex*)
- 5.279.2.12 PUBLIC_FUNC_DECL (char *, *hex_encode*, const unsigned char * *buf*, size_t *len*)
- 5.279.2.13 PUBLIC_FUNC_DECL (char *, *b64encode_w_line separators*, const unsigned char * *buf*, size_t *len*)
- 5.279.2.14 PUBLIC_FUNC_DECL (char *, *b64encode_nopad*, const unsigned char * *buf*, size_t *len*)
- 5.279.2.15 PUBLIC_FUNC_DECL (char *, *b64encode*, const unsigned char * *buf*, size_t *len*)
- 5.279.2.16 PUBLIC_FUNC_DECL (unsigned char *, *b64decode_nopad*, const char * *buf*, size_t *len*, size_t * *outlen*)
- 5.279.2.17 PUBLIC_FUNC_DECL (unsigned char *, *b64decode*, const char * *buf*, size_t *len*, size_t * *outlen*)
- 5.279.2.18 PUBLIC_FUNC_DECL (void, *secure_wipe*, void * *buf*, size_t *len*)
- 5.279.2.19 PUBLIC_FUNC_DECL (int, *is_buf_zeroed*, void * *buf*, size_t *len*)
- 5.279.2.20 PUBLIC_FUNC_DECL (char *, *get_chr_date*, time_t *time*, int *local*)
- 5.279.2.21 PUBLIC_FUNC_DECL (void *, *ptr_chain_clone*, void * *buf*)
- 5.279.2.22 PUBLIC_FUNC_DECL (int, *count_ptr_chain*, void * *buf*)
- 5.279.2.23 PUBLIC_FUNC_DECL (void, *ptr_chain_free*, void * *buf*)
- 5.279.2.24 PUBLIC_FUNC_DECL (void *, *ptr_chain_add*, void * *buf*, const void * *addr*)
- 5.279.2.25 PUBLIC_FUNC_DECL (size_t, *mem_append*, unsigned char ** *buf*, size_t * *blen*, const unsigned char * *data*, size_t *dlen*)
- 5.279.2.26 PUBLIC_FUNC_DECL (void, *int_no_put_2b*, void * *buf*, uint16_t *val*)
- 5.279.2.27 PUBLIC_FUNC_DECL (void, *int_no_put_3b*, void * *buf*, uint32_t *val*)
- 5.279.2.28 PUBLIC_FUNC_DECL (void, *int_no_put_4b*, void * *buf*, uint32_t *val*)

Referenced by `__dbgprint()`, `_add_cached_object_cmp_forced()`, `_add_cached_object_forced()`, `_do_ocsp_validation()`, `_do_x509_validation()`, `_get_dbg_level()`, `_get_txt_record()`, `_load_cache_contents()`, `_mem_append()`, `_ocsp_response_callback()`, `_set_dbg_level()`, `_unlink_object()`, and `_validate_rrsig_rr()`.

5.280 src/providers/dime/common/misc_pub.c File Reference

```
#include "dime/common/misc.h"
#include "dime/common/error.h"
```

Functions

- void [int_no_put_4b](#) (void *buf, uint32_t val)
- void [int_no_put_3b](#) (void *buf, uint32_t val)
- void [int_no_put_2b](#) (void *buf, uint16_t val)
- uint32_t [int_no_get_4b](#) (const void *buf)
- uint32_t [int_no_get_3b](#) (const void *buf)
- uint16_t [int_no_get_2b](#) (const void *buf)
- char * [b64encode](#) (const unsigned char *buf, size_t len)
- char * [b64encode_nopad](#) (const unsigned char *buf, size_t len)
- unsigned char * [b64decode](#) (const char *buf, size_t len, size_t *outlen)
- unsigned char * [b64decode_nopad](#) (const char *buf, size_t len, size_t *outlen)
- char * [hex_encode](#) (const unsigned char *buf, size_t len)
- void [set_dbg_level](#) (unsigned int level)
- unsigned int [get_dbg_level](#) (void)
- int [str_printf](#) (char **sbuf, const char *fmt,...)
- size_t [mem_append](#) (unsigned char **buf, size_t *blen, const unsigned char *data, size_t dlen)
- void [ptr_chain_free](#) (void *buf)
- void * [ptr_chain_add](#) (void *buf, const void *addr)
- int [count_ptr_chain](#) (void *buf)
- void * [ptr_chain_clone](#) (void *buf)
- char * [get_chr_date](#) (time_t time, int local)
- void [dump_buf](#) (const unsigned char *buf, size_t len, int all_hex)
- void [dump_buf_outer](#) (const unsigned char *buf, size_t len, size_t nouter, int all_hex)
- void [dbgprint](#) (unsigned int dbglevel, const char *fmt,...)
- int [compute_sha_hash](#) (size_t nbits, const unsigned char *buf, size_t blen, unsigned char *outbuf)
- int [compute_sha_hash_multibuf](#) (size_t nbits, [sha_databuf_t](#) *bufs, unsigned char *outbuf)
- RSA * [decode_rsa_pubkey](#) (unsigned char *data, size_t dlen)
- unsigned char * [encode_rsa_pubkey](#) (RSA *pubkey, size_t *enclen)
- int [get_x509_cert_sha_hash](#) (X509 *cert, size_t nbits, unsigned char *out)
- int [is_buf_zeroed](#) (void *buf, size_t len)
- unsigned char * [read_file_data](#) (const char *filename, size_t *fsize)
- char * [read_pem_data](#) (const char *pemfile, const char *tag, int nospace)
- void [secure_wipe](#) (void *buf, size_t len)

5.280.1 Function Documentation

5.280.1.1 unsigned char* b64decode (const char * buf, size_t len, size_t * outlen)

Definition at line 35 of file misc_pub.c.

References [PUBLIC_FUNC_IMPL](#).

5.280.1.2 unsigned char* b64decode_nopad (const char * buf, size_t len, size_t * outlen)

Definition at line 39 of file misc_pub.c.

References [PUBLIC_FUNC_IMPL](#).

5.280.1.3 char* b64encode (const unsigned char * *buf*, size_t *len*)

Definition at line 28 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.4 char* b64encode_nopad (const unsigned char * *buf*, size_t *len*)

Definition at line 31 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.5 int compute_sha_hash (size_t *nbits*, const unsigned char * *buf*, size_t *blen*, unsigned char * *outbuf*)

Definition at line 95 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.6 int compute_sha_hash_multibuf (size_t *nbits*, sha_databuf_t * *bufs*, unsigned char * *outbuf*)

Definition at line 99 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.7 int count_ptr_chain (void * *buf*)

Definition at line 71 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.8 void dbgprint (unsigned int *dbglevel*, const char * *fmt*, ...)

Definition at line 91 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VA2.

Referenced by _read_pem_data().

5.280.1.9 RSA* decode_rsa_pubkey (unsigned char * *data*, size_t *dlen*)

Definition at line 103 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.10 void dump_buf (const unsigned char * *buf*, size_t *len*, int *all_hex*)

Definition at line 83 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.280.1.11 void dump_buf_outer (const unsigned char * *buf*, size_t *len*, size_t *nrouter*, int *all_hex*)

Definition at line 87 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.280.1.12 unsigned char* encode_rsa_pubkey (RSA * pubkey, size_t * enclen)

Definition at line 107 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.13 char* get_chr_date (time_t time, int local)

Definition at line 79 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.14 unsigned int get_dbg_level (void)

Definition at line 51 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.15 int get_x509_cert_sha_hash (X509 * cert, size_t nbits, unsigned char * out)

Definition at line 111 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.16 char* hex_encode (const unsigned char * buf, size_t len)

Definition at line 43 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.17 uint16_t int_no_get_2b (const void * buf)

Definition at line 24 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.18 uint32_t int_no_get_3b (const void * buf)

Definition at line 20 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.19 uint32_t int_no_get_4b (const void * buf)

Definition at line 16 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.20 void int_no_put_2b (void * buf, uint16_t val)

Definition at line 12 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.280.1.21 void int_no_put_3b (void * *buf*, uint32_t *val*)

Definition at line 8 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.280.1.22 void int_no_put_4b (void * *buf*, uint32_t *val*)

Definition at line 4 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.280.1.23 int is_buf_zeroed (void * *buf*, size_t *len*)

Definition at line 115 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.24 size_t mem_append (unsigned char ** *buf*, size_t * *blen*, const unsigned char * *data*, size_t *dlen*)

Definition at line 59 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.25 void* ptr_chain_add (void * *buf*, const void * *addr*)

Definition at line 67 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.26 void* ptr_chain_clone (void * *buf*)

Definition at line 75 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.27 void ptr_chain_free (void * *buf*)

Definition at line 63 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.280.1.28 unsigned char* read_file_data (const char * *filename*, size_t * *fsize*)

Definition at line 119 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.29 char* read_pem_data (const char * *pemfile*, const char * *tag*, int *nospace*)

Definition at line 123 of file misc_pub.c.

References PUBLIC_FUNC_IMPL.

5.280.1.30 void secure_wipe (void * *buf*, size_t *len*)

Definition at line 127 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.280.1.31 void set_dbg_level (unsigned int *level*)

Definition at line 47 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.280.1.32 int str_printf (char ** *sbuf*, const char * *fmt*, ...)

Definition at line 55 of file misc_pub.c.

References PUBLIC_FUNC_IMPL_VA2_RET.

Referenced by _get_dime_dir_location().

5.281 src/providers/dime/common/network.c File Reference

```
#include <sys/types.h>
#include <sys/socket.h>
#include <errno.h>
#include "dime/common/network.h"
#include "dime/common/misc.h"
```

Defines

- #define [CONNECT_TIMEOUT](#) 5

Functions

- int [_connect_host](#) (char const *hostname, unsigned short port, int force_family)
Connect to a host/tcp port in an address-independent manner, and return a file descriptor.
- int [_connect_timeout](#) (int fd, struct sockaddr const *addr, socklen_t addrlen)
Attempt a TCP connection with a time-out mechanism.

5.281.1 Define Documentation

5.281.1.1 #define CONNECT_TIMEOUT 5

Definition at line 10 of file network.c.

Referenced by [_connect_timeout\(\)](#).

5.281.2 Function Documentation

5.281.2.1 int _connect_host (char const * *hostname*, unsigned short *port*, int *force_family*)

Connect to a host/tcp port in an address-independent manner, and return a file descriptor.

Parameters:

hostname the hostname of the server to connect to.

port the TCP port number to which the connection should be made.

force_family an optional address family to force the connection to (AF_INET or AF_INET6), or 0 to ignore).

Returns:

-1 on general failure or the file descriptor of the socket connection on success.

Definition at line 28 of file network.c.

References [_connect_timeout\(\)](#), [_dbgprint\(\)](#), [ERR_BAD_PARAM](#), [ERR_UNSPEC](#), [RET_ERROR_INT](#), and [RET_ERROR_INT_FMT](#).

Referenced by [_do_ocsp_validation\(\)](#), [_dx_connect_dual\(\)](#), and [_ssl_connect_host\(\)](#).

5.281.2.2 `int _connect_timeout (int fd, struct sockaddr const * addr, socklen_t addrlen)`

Attempt a TCP connection with a time-out mechanism.

Parameters:

fd the open socket descriptor across which the connection will be attempted.

addr a pointer to the prepared sockaddr structure that is the target of the connection.

addrlen the size of the supplied sockaddr structure.

Returns:

-1 on general error, 0 if the connection failed or timed out, and 1 on success.

Definition at line 113 of file network.c.

References `CONNECT_TIMEOUT`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_INT`.

Referenced by `_connect_host()`.

5.282 src/providers/dime/common/network_pub.c File Reference

```
#include "dime/common/network.h"
#include "dime/common/error.h"
```

Functions

- int [connect_host](#) (const char *hostname, unsigned short port, int force_family)

5.282.1 Function Documentation

5.282.1.1 int connect_host (const char * *hostname*, unsigned short *port*, int *force_family*)

Definition at line 4 of file network_pub.c.

References PUBLIC_FUNC_IMPL.

5.283 src/providers/dime/dime_ctx.c File Reference

```
#include "dime/dime_ctx.h"
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
```

Data Structures

- struct [dime_ctx](#)

Functions

- [derror_t](#) const * [dime_ctx_new](#) ([dime_ctx_t](#) **result, [log_function_t](#) log_callback)
- void [dime_ctx_free](#) ([dime_ctx_t](#) *ctx)
- void [dime_ctx_log](#) ([dime_ctx_t](#) const *ctx, char const *file, size_t line, [log_level_t](#) const *level, char const *format,...)

Variables

- [log_level_t](#) const *const [LOG_LEVEL_DEBUG](#) = &DEBUG_LOG_LEVEL
- [log_level_t](#) const *const [LOG_LEVEL_INFO](#) = &INFO_LOG_LEVEL
- [log_level_t](#) const *const [LOG_LEVEL_ERROR](#) = &ERROR_LOG_LEVEL

5.283.1 Function Documentation

5.283.1.1 void dime_ctx_free (dime_ctx_t * ctx)

Definition at line 90 of file dime_ctx.c.

5.283.1.2 void dime_ctx_log (dime_ctx_t const * ctx, char const * file, size_t line, log_level_t const * level, char const * format, ...)

Definition at line 97 of file dime_ctx.c.

References [dime_ctx::log_callback](#).

5.283.1.3 derror_t const* dime_ctx_new (dime_ctx_t ** result, log_function_t log_callback)

Definition at line 62 of file dime_ctx.c.

References [ERR_BAD_PARAM](#), and [ERR_NOMEM](#).

5.283.2 Variable Documentation

5.283.2.1 log_level_t const* const LOG_LEVEL_DEBUG = &DEBUG_LOG_LEVEL

Definition at line 12 of file dime_ctx.c.

5.283.2.2 log_level_t const* const LOG_LEVEL_ERROR = &ERROR_LOG_LEVEL

Definition at line 26 of file dime_ctx.c.

5.283.2.3 `log_level_t` `const*` `const` `LOG_LEVEL_INFO` = `&INFO_LOG_LEVEL`

Definition at line 19 of file `dime_ctx.c`.

5.284 src/providers/dime/dime_ctx.h File Reference

```
#include "dime/error_codes.h"
#include "providers/symbols.h"
#include <stddef.h>
#include <stdarg.h>
```

Data Structures

- struct [log_level_t](#)

Defines

- #define [DIME_LOG_DEBUG](#)(ctx,...)
- #define [DIME_LOG_INFO](#)(ctx,...)
- #define [DIME_LOG_ERROR](#)(ctx,...)

Typedefs

- typedef void(* [log_function_t](#))(char const *file, size_t line, [log_level_t](#) const *level, char const *format, va_list argp)
- typedef struct [dime_ctx](#) [dime_ctx_t](#)

Enumerations

- enum [log_code_t](#) { [LOG_CODE_DEBUG](#), [LOG_CODE_INFO](#), [LOG_CODE_ERROR](#) }

Functions

- [derror_t](#) const * [dime_ctx_new](#) ([dime_ctx_t](#) **result, [log_function_t](#) log_callback)
- void [dime_ctx_free](#) ([dime_ctx_t](#) *ctx)
- void [dime_ctx_log](#) ([dime_ctx_t](#) const *ctx, char const *file, size_t line, [log_level_t](#) const *level, char const *format,...)

Variables

- [log_level_t](#) const *const [LOG_LEVEL_DEBUG](#)
- [log_level_t](#) const *const [LOG_LEVEL_INFO](#)
- [log_level_t](#) const *const [LOG_LEVEL_ERROR](#)

5.284.1 Define Documentation

5.284.1.1 #define DIME_LOG_DEBUG(ctx, ...)

Value:

```
dime_ctx_log( \
    ctx, \
    __FILE__, \
    __LINE__, \
    LOG_LEVEL_DEBUG, \
    __VA_ARGS__)
```

Definition at line 58 of file `dime_ctx.h`.

5.284.1.2 #define DIME_LOG_ERROR(ctx, ...)

Value:

```
dime_ctx_log( \
    ctx, \
    __FILE__, \
    __LINE__, \
    LOG_LEVEL_ERROR, \
    __VA_ARGS__)
```

Definition at line 74 of file dime_ctx.h.

Referenced by libdime_base64_decode(), and libdime_base64_encode().

5.284.1.3 #define DIME_LOG_INFO(ctx, ...)

Value:

```
dime_ctx_log( \
    ctx, \
    __FILE__, \
    __LINE__, \
    LOG_LEVEL_INFO, \
    __VA_ARGS__)
```

Definition at line 66 of file dime_ctx.h.

5.284.2 Typedef Documentation

5.284.2.1 typedef struct dime_ctx dime_ctx_t

Definition at line 38 of file dime_ctx.h.

5.284.2.2 typedef void(* log_function_t)(char const *file, size_t line, log_level_t const *level, char const *format, va_list argp)

Definition at line 30 of file dime_ctx.h.

5.284.3 Enumeration Type Documentation

5.284.3.1 enum log_code_t

Enumerator:

```
LOG_CODE_DEBUG
LOG_CODE_INFO
LOG_CODE_ERROR
```

Definition at line 10 of file dime_ctx.h.

5.284.4 Function Documentation

5.284.4.1 void dime_ctx_free (dime_ctx_t * ctx)

Definition at line 90 of file dime_ctx.c.

5.284.4.2 `void dime_ctx_log (dime_ctx_t const * ctx, char const * file, size_t line, log_level_t const * level, char const * format, ...)`

Definition at line 97 of file `dime_ctx.c`.

References `dime_ctx::log_callback`.

5.284.4.3 `derror_t const* dime_ctx_new (dime_ctx_t ** result, log_function_t log_callback)`

Definition at line 62 of file `dime_ctx.c`.

References `ERR_BAD_PARAM`, and `ERR_NOMEM`.

5.284.5 Variable Documentation

5.284.5.1 `log_level_t const* const LOG_LEVEL_DEBUG`

Definition at line 12 of file `dime_ctx.c`.

5.284.5.2 `log_level_t const* const LOG_LEVEL_ERROR`

Definition at line 26 of file `dime_ctx.c`.

5.284.5.3 `log_level_t const* const LOG_LEVEL_INFO`

Definition at line 19 of file `dime_ctx.c`.

5.285 src/providers/dime/dmessage/common.h File Reference

```
#include "dime/sds/sds.h"
```

Data Structures

- struct [dmime_common_headers_t](#)
- struct [dmime_chunk_key_t](#)

Defines

- #define [DMIME_NUM_COMMON_HEADERS](#) 7
- #define [CHUNK_LENGTH_SIZE](#) 3
- #define [TRACING_HEADER_SIZE](#) 4
- #define [MESSAGE_LENGTH_SIZE](#) 4
- #define [MESSAGE_HEADER_SIZE](#) 6
- #define [CHUNK_HEADER_SIZE](#) 4
- #define [DMIME_CHUNK_TYPE_MAX](#) 256
- #define [MINIMUM_PAYLOAD_SIZE](#) 256
- #define [ALTERNATE_PADDING_ALGORITHM_ENABLED](#) 1
- #define [ALTERNATE_USER_KEY_APPLIED_TO_DATE](#) 2
- #define [GZIP_COMPRESSION_ENABLED](#) 4
- #define [DATA_SEGMENT_CONTINUATION_ENABLED](#) 128
- #define [DEFAULT_CHUNK_FLAGS](#) 0

Enumerations

- enum [dmime_bounce_type_t](#) { [META_BOUNCE](#) = 1, [DISPLAY_BOUNCE](#) }
- enum [dmime_chunk_type_t](#) {
[CHUNK_TYPE_NONE](#) = 0, [CHUNK_TYPE_EPHEMERAL](#) = 2, [CHUNK_TYPE_ALTERNATE](#), [CHUNK_TYPE_ORIGIN](#),
[CHUNK_TYPE_DESTINATION](#), [CHUNK_TYPE_META_COMMON](#) = 33, [CHUNK_TYPE_META_OTHER](#), [CHUNK_TYPE_DISPLAY_CONTENT](#) = 67,
[CHUNK_TYPE_ATTACH_CONTENT](#) = 131, [CHUNK_TYPE_SIG_AUTHOR_TREE](#) = 225, [CHUNK_TYPE_SIG_AUTHOR_FULL](#) = 226, [CHUNK_TYPE_SIG_ORIGIN_META_BOUNCE](#) = 248,
[CHUNK_TYPE_SIG_ORIGIN_DISPLAY_BOUNCE](#) = 249, [CHUNK_TYPE_SIG_ORIGIN_FULL](#) = 255 }
- enum [dmime_chunk_section_t](#) {
[CHUNK_SECTION_NONE](#) = 0, [CHUNK_SECTION_ENVELOPE](#) = 1, [CHUNK_SECTION_METADATA](#) = 2, [CHUNK_SECTION_DISPLAY](#) = 4,
[CHUNK_SECTION_ATTACH](#) = 8, [CHUNK_SECTION_SIG](#) = 16 }
- enum [dmime_payload_type_t](#) { [PAYLOAD_TYPE_NONE](#) = 0, [PAYLOAD_TYPE_EPHEMERAL](#), [PAYLOAD_TYPE_STANDARD](#), [PAYLOAD_TYPE_SIGNATURE](#) }

Variables

- [dmime_chunk_key_t dmime_chunk_keys](#) [[DMIME_CHUNK_TYPE_MAX](#)]

5.285.1 Define Documentation

5.285.1.1 #define ALTERNATE_PADDING_ALGORITHM_ENABLED 1

Definition at line 17 of file common.h.

5.285.1.2 #define ALTERNATE_USER_KEY_APPLIED_TO_DATE 2

Definition at line 18 of file common.h.

5.285.1.3 #define CHUNK_HEADER_SIZE 4

Definition at line 11 of file common.h.

5.285.1.4 #define CHUNK_LENGTH_SIZE 3

Definition at line 7 of file common.h.

Referenced by __attribute__().

5.285.1.5 #define DATA_SEGMENT_CONTINUATION_ENABLED 128

Definition at line 20 of file common.h.

5.285.1.6 #define DEFAULT_CHUNK_FLAGS 0

Definition at line 22 of file common.h.

5.285.1.7 #define DMIME_CHUNK_TYPE_MAX 256

Definition at line 13 of file common.h.

5.285.1.8 #define DMIME_NUM_COMMON_HEADERS 7

Definition at line 6 of file common.h.

5.285.1.9 #define GZIP_COMPRESSION_ENABLED 4

Definition at line 19 of file common.h.

5.285.1.10 #define MESSAGE_HEADER_SIZE 6

Definition at line 10 of file common.h.

5.285.1.11 #define MESSAGE_LENGTH_SIZE 4

Definition at line 9 of file common.h.

5.285.1.12 #define MINIMUM_PAYLOAD_SIZE 256

Definition at line 15 of file common.h.

5.285.1.13 #define TRACING_HEADER_SIZE 4

Definition at line 8 of file common.h.

5.285.2 Enumeration Type Documentation

5.285.2.1 enum dmime_bounce_type_t

Enumerator:

META_BOUNCE

DISPLAY_BOUNCE

Definition at line 28 of file common.h.

5.285.2.2 enum dmime_chunk_section_t

Enumerator:

CHUNK_SECTION_NONE

CHUNK_SECTION_ENVELOPE

CHUNK_SECTION_METADATA

CHUNK_SECTION_DISPLAY

CHUNK_SECTION_ATTACH

CHUNK_SECTION_SIG

Definition at line 100 of file common.h.

5.285.2.3 enum dmime_chunk_type_t

Enumerator:

CHUNK_TYPE_NONE

CHUNK_TYPE_EPHEMERAL

CHUNK_TYPE_ALTERNATE

CHUNK_TYPE_ORIGIN

CHUNK_TYPE_DESTINATION

CHUNK_TYPE_META_COMMON

CHUNK_TYPE_META_OTHER

CHUNK_TYPE_DISPLAY_CONTENT

CHUNK_TYPE_ATTACH_CONTENT

CHUNK_TYPE_SIG_AUTHOR_TREE

CHUNK_TYPE_SIG_AUTHOR_FULL

CHUNK_TYPE_SIG_ORIGIN_META_BOUNCE

CHUNK_TYPE_SIG_ORIGIN_DISPLAY_BOUNCE

CHUNK_TYPE_SIG_ORIGIN_FULL

Definition at line 35 of file common.h.

5.285.2.4 enum dmime_payload_type_t

Enumerator:

PAYLOAD_TYPE_NONE

PAYLOAD_TYPE_EPHEMERAL

PAYLOAD_TYPE_STANDARD

PAYLOAD_TYPE_SIGNATURE

Definition at line 111 of file common.h.

5.285.3 Variable Documentation

5.285.3.1 dmime_chunk_key_t dmime_chunk_keys[DMIME_CHUNK_TYPE_MAX]

Definition at line 6491 of file dmsg.c.

5.286 src/providers/dime/signet/common.h File Reference

```
#include <stdint.h>
#include "dime/common/dcrypto.h"
```

Data Structures

- struct [signet_field_key_t](#)

Defines

- #define [SIGNET_VER_NO](#) 0x1
- #define [SIGNET_HEADER_SIZE](#) 5
- #define [SIGNET_KEY_ORG](#) "ORGANIZATIONAL KEY"
- #define [SIGNET_KEY_USER](#) "USER KEY"
- #define [SIGNET_ORG](#) "ORGANIZATIONAL SIGNET"
- #define [SIGNET_USER](#) "USER SIGNET"
- #define [KEYS_HEADER_SIZE](#) 5
- #define [FIELD_NAME_MAX_SIZE](#) 255
- #define [UNSIGNED_MAX_1_BYTE](#) 255
- #define [UNSIGNED_MAX_2_BYTE](#) 65535
- #define [UNSIGNED_MAX_3_BYTE](#) 16777215
- #define [SIGNET_FID_MAX](#) 255
- #define [KEYS_FID_MAX](#) 3
- #define [DIME_NUMBER_SIZE](#) 2
- #define [SIGNET_MAX_SIZE](#) 16777220

Enumerations

- enum [dime_number_t](#) {
[DIME_SSR](#) = 1215, [DIME_ORG_SIGNET](#) = 1776, [DIME_USER_SIGNET](#) = 1789, [DIME_ENCRYPTED_ORG_KEYS](#) = 1947,
[DIME_ORG_KEYS](#) = 1952, [DIME_ENCRYPTED_USER_KEYS](#) = 1976, [DIME_USER_KEYS](#) = 2013, [DIME_MSG_TRACING](#) =
1837,
[DIME_ENCRYPTED_MSG](#) = 1847 }
- enum [sok_permissions_t](#) {
[SIGNET_SOK_NONE](#) = 0, [SIGNET_SOK_SIGNET](#) = 1, [SIGNET_SOK_MSG](#) = 2, [SIGNET_SOK_TLS](#) = 4,
[SIGNET_SOK_SOFTWARE](#) = 8 }
- enum [signet_org_field_t](#) {
[SIGNET_ORG_POK](#) = 1, [SIGNET_ORG_SOK](#), [SIGNET_ORG_ENC_KEY](#), [SIGNET_ORG_CRYPTOSIG](#),
[SIGNET_ORG_NAME](#) = 16, [SIGNET_ORG_ADDRESS](#), [SIGNET_ORG_PROVINCE](#), [SIGNET_ORG_COUNTRY](#),
[SIGNET_ORG_POSTAL](#), [SIGNET_ORG_PHONE](#), [SIGNET_ORG_LANGUAGE](#), [SIGNET_ORG_CURRENCY](#),
[SIGNET_ORG_CRYPTOCURRENCY](#), [SIGNET_ORG_MOTTO](#), [SIGNET_ORG_EXTENSIONS](#), [SIGNET_ORG_MSG_SIZE](#) -
LIM,
[SIGNET_ORG_WEBSITE](#) = 160, [SIGNET_ORG_ABUSE](#) = 200, [SIGNET_ORG_ADMIN](#), [SIGNET_ORG_SUPPORT](#),
[SIGNET_ORG_WEB_HOST](#), [SIGNET_ORG_WEB_LOCATION](#), [SIGNET_ORG_WEB_CERT](#), [SIGNET_ORG_MAIL_HOST](#),
[SIGNET_ORG_MAIL_CERT](#), [SIGNET_ORG_ONION_ACCESS_HOST](#), [SIGNET_ORG_ONION_ACCESS_CERT](#), [SIGNET](#) -
[ORG_ONION_DELIVERY_HOST](#),
[SIGNET_ORG_ONION_DELIVERY_CERT](#), [SIGNET_ORG_UNDEFINED](#) = 251, [SIGNET_ORG_PHOTO](#), [SIGNET_ORG](#) -
[FULL_SIG](#),
[SIGNET_ORG_ID](#), [SIGNET_ORG_ID_SIG](#) }

- enum `signet_user_field_t` {
`SIGNET_USER_SIGN_KEY = 1`, `SIGNET_USER_ENC_KEY`, `SIGNET_USER_ALT_KEY`, `SIGNET_USER_COC_SIG`,
`SIGNET_USER_SSR_SIG`, `SIGNET_USER_CRYPTO_SIG`, `SIGNET_USER_NAME = 16`, `SIGNET_USER_ADDRESS`,
`SIGNET_USER_PROVINCE`, `SIGNET_USER_COUNTRY`, `SIGNET_USER_POSTAL`, `SIGNET_USER_PHONE`,
`SIGNET_USER_LANGUAGE`, `SIGNET_USER_CURRENCY`, `SIGNET_USER_CRYPTOCURRENCY`, `SIGNET_USER_MOTTO`,
`SIGNET_USER_EXTENSIONS`, `SIGNET_USER_MSG_SIZE_LIM`, `SIGNET_USER_CODECS = 93`, `SIGNET_USER_TITLE`,
`SIGNET_USER_EMPLOYER`, `SIGNET_USER_GENDER`, `SIGNET_USER_ALMA_MATER`, `SIGNET_USER_SUPERVISOR`,
`SIGNET_USER_POLITICAL_PARTY`, `SIGNET_USER_ALTERNATE_ADDRESS = 200`, `SIGNET_USER_RESUME`, `SIGNET_USER_ENDORSEMENTS`,
`SIGNET_USER_UNDEFINED = 251`, `SIGNET_USER_PHOTO`, `SIGNET_USER_FULL_SIG`, `SIGNET_USER_ID`,
`SIGNET_USER_ID_SIG` }
- enum `signet_ssr_field_t` {
`SIGNET_SSR_SIGN_KEY = 1`, `SIGNET_SSR_ENC_KEY`, `SIGNET_SSR_ALT_KEY`, `SIGNET_SSR_COC_SIG`,
`SIGNET_SSR_SSR_SIG` }
- enum `keys_org_t` { `KEYS_ORG_PRIVATE_POK = 1`, `KEYS_ORG_PRIVATE_SOK`, `KEYS_ORG_PRIVATE_ENC` }
- enum `keys_user_t` { `KEYS_USER_PRIVATE_SIGN = 1`, `KEYS_USER_PRIVATE_ENC` }
- enum `signkey_format_t` { `SIGNKEY_DEFAULT_FORMAT = 0x40` }
- enum `field_data_t` { `B64`, `HEX`, `PNG`, `UNICODE` }

Functions

- const char * `dime_number_to_str` (`dime_number_t` number)

Returns a string from a `dime_number_t` enum type.

Variables

- `signet_field_key_t` `signet_org_field_keys` [256]
- `signet_field_key_t` `signet_user_field_keys` [256]
- `signet_field_key_t` `signet_ssr_field_keys` [256]

5.286.1 Define Documentation

5.286.1.1 #define DIME_NUMBER_SIZE 2

Definition at line 20 of file `common.h`.

5.286.1.2 #define FIELD_NAME_MAX_SIZE 255

Definition at line 14 of file `common.h`.

5.286.1.3 #define KEYS_FID_MAX 3

Definition at line 19 of file `common.h`.

5.286.1.4 #define KEYS_HEADER_SIZE 5

Definition at line 13 of file `common.h`.

5.286.1.5 #define SIGNET_FID_MAX 255

Definition at line 18 of file common.h.

5.286.1.6 #define SIGNET_HEADER_SIZE 5

Definition at line 8 of file common.h.

5.286.1.7 #define SIGNET_KEY_ORG "ORGANIZATIONAL KEY"

Definition at line 9 of file common.h.

5.286.1.8 #define SIGNET_KEY_USER "USER KEY"

Definition at line 10 of file common.h.

5.286.1.9 #define SIGNET_MAX_SIZE 16777220

Definition at line 23 of file common.h.

5.286.1.10 #define SIGNET_ORG "ORGANIZATIONAL SIGNET"

Definition at line 11 of file common.h.

5.286.1.11 #define SIGNET_USER "USER SIGNET"

Definition at line 12 of file common.h.

5.286.1.12 #define SIGNET_VER_NO 0x1

Definition at line 7 of file common.h.

5.286.1.13 #define UNSIGNED_MAX_1_BYTE 255

Definition at line 15 of file common.h.

5.286.1.14 #define UNSIGNED_MAX_2_BYTE 65535

Definition at line 16 of file common.h.

5.286.1.15 #define UNSIGNED_MAX_3_BYTE 16777215

Definition at line 17 of file common.h.

5.286.2 Enumeration Type Documentation

5.286.2.1 enum dime_number_t

Enumerator:

DIME_SSR Dime numbers are the magic numbers File contains an ssr

DIME_ORG_SIGNET File contains an organizational signet

DIME_USER_SIGNET File contains a user signet

DIME_ENCRYPTED_ORG_KEYS File contains an encrypted organizational key.

DIME_ORG_KEYS File contains organizational keys

DIME_ENCRYPTED_USER_KEYS File contains an encrypted user key.

DIME_USER_KEYS File contains user keys

DIME_MSG_TRACING

DIME_ENCRYPTED_MSG

Definition at line 25 of file common.h.

5.286.2.2 enum field_data_t

Enumerator:

B64 Currently barely used, meant to classify signet field data types

HEX

PNG

UNICODE

Definition at line 143 of file common.h.

5.286.2.3 enum keys_org_t

Enumerator:

KEYS_ORG_PRIVATE_POK

KEYS_ORG_PRIVATE_SOK

KEYS_ORG_PRIVATE_ENC

Definition at line 126 of file common.h.

5.286.2.4 enum keys_user_t

Enumerator:

KEYS_USER_PRIVATE_SIGN

KEYS_USER_PRIVATE_ENC

Definition at line 132 of file common.h.

5.286.2.5 enum signet_org_field_t**Enumerator:**

SIGNET_ORG_POK The ed25519 public signing key of the signet holder
SIGNET_ORG_SOK Secondary Organization Signing keys
SIGNET_ORG_ENC_KEY The ECC public encryption key of the signet holder
SIGNET_ORG_CRYPTOSIG Org signature of all previous fields
SIGNET_ORG_NAME
SIGNET_ORG_ADDRESS
SIGNET_ORG_PROVINCE
SIGNET_ORG_COUNTRY
SIGNET_ORG_POSTAL
SIGNET_ORG_PHONE
SIGNET_ORG_LANGUAGE
SIGNET_ORG_CURRENCY
SIGNET_ORG_CRYPTOCURRENCY
SIGNET_ORG_MOTTO
SIGNET_ORG_EXTENSIONS
SIGNET_ORG_MSG_SIZE_LIM
SIGNET_ORG_WEBSITE
SIGNET_ORG_ABUSE
SIGNET_ORG_ADMIN
SIGNET_ORG_SUPPORT
SIGNET_ORG_WEB_HOST
SIGNET_ORG_WEB_LOCATION
SIGNET_ORG_WEB_CERT
SIGNET_ORG_MAIL_HOST
SIGNET_ORG_MAIL_CERT
SIGNET_ORG_ONION_ACCESS_HOST
SIGNET_ORG_ONION_ACCESS_CERT
SIGNET_ORG_ONION_DELIVERY_HOST
SIGNET_ORG_ONION_DELIVERY_CERT
SIGNET_ORG_UNDEFINED Unicode undefined field
SIGNET_ORG_PHOTO Organizational photo
SIGNET_ORG_FULL_SIG ORG signature
SIGNET_ORG_ID Org Signet ID
SIGNET_ORG_ID_SIG Org Signature following the ID field

Definition at line 45 of file common.h.

5.286.2.6 enum signet_ssr_field_t**Enumerator:**

SIGNET_SSR_SIGN_KEY The proposed ed25519 public signing key of the ssr creator
SIGNET_SSR_ENC_KEY The ed25519 ECC public encryption key of the ssr creator
SIGNET_SSR_ALT_KEY Alternative encryption keys for the ssr creator
SIGNET_SSR_COC_SIG Chain of custody signature by user's previous signing key
SIGNET_SSR_SSR_SIG User signature with user's signing key

Definition at line 118 of file common.h.

5.286.2.7 enum signet_user_field_t**Enumerator:**

SIGNET_USER_SIGN_KEY The ed25519 public signing key of the signet holder
SIGNET_USER_ENC_KEY The ECC public encryption key of the signet holder
SIGNET_USER_ALT_KEY Alternative encryption keys for the user
SIGNET_USER_COC_SIG Chain of custody signature by user's previous signing key
SIGNET_USER_SSR_SIG User signature with user's signing key
SIGNET_USER_CRYPTO_SIG Initial signature by the organization's signing key
SIGNET_USER_NAME
SIGNET_USER_ADDRESS
SIGNET_USER_PROVINCE
SIGNET_USER_COUNTRY
SIGNET_USER_POSTAL
SIGNET_USER_PHONE
SIGNET_USER_LANGUAGE
SIGNET_USER_CURRENCY
SIGNET_USER_CRYPTOCURRENCY
SIGNET_USER_MOTTO
SIGNET_USER_EXTENSIONS
SIGNET_USER_MSG_SIZE_LIM
SIGNET_USER_CODECS
SIGNET_USER_TITLE
SIGNET_USER_EMPLOYER
SIGNET_USER_GENDER
SIGNET_USER_ALMA_MATER
SIGNET_USER_SUPERVISOR
SIGNET_USER_POLITICAL_PARTY
SIGNET_USER_ALTERNATE_ADDRESS
SIGNET_USER_RESUME
SIGNET_USER_ENDORSEMENTS
SIGNET_USER_UNDEFINED ASCII undefined field
SIGNET_USER_PHOTO User photo
SIGNET_USER_FULL_SIG Final Organizational Signature
SIGNET_USER_ID User Signet ID
SIGNET_USER_ID_SIG Org Signature following the ID field

Definition at line 82 of file common.h.

5.286.2.8 enum signkey_format_t**Enumerator:**

SIGNKEY_DEFAULT_FORMAT Currently the only legal format specifier for ED25519 signing keys

Definition at line 137 of file common.h.

5.286.2.9 enum sok_permissions_t

Enumerator:

SIGNET_SOK_NONE SOK = Secondary Organizational Key Can not be used for signing anything

SIGNET_SOK_SIGNET Can be used for signing signets

SIGNET_SOK_MSG Can be used for signing messages

SIGNET_SOK_TLS Can be used for signing TLS certificates

SIGNET_SOK_SOFTWARE Can be used for signing software

Definition at line 37 of file common.h.

5.286.3 Function Documentation

5.286.3.1 const char* dime_number_to_str (dime_number_t *number*)

Returns a string from a dime_number_t enum type.

Parameters:

number Dime number input.

Returns:

Null terminated string corresponding to the dime number.

Definition at line 1070 of file general.c.

References DIME_ENCRYPTED_MSG, DIME_ENCRYPTED_ORG_KEYS, DIME_ENCRYPTED_USER_KEYS, DIME_MSG-TRACING, DIME_ORG_KEYS, DIME_ORG_SIGNET, DIME_SSR, DIME_USER_KEYS, and DIME_USER_SIGNET.

5.286.4 Variable Documentation

5.286.4.1 signet_field_key_t signet_org_field_keys[256]

Definition at line 18 of file general.c.

5.286.4.2 signet_field_key_t signet_ssr_field_keys[256]

Definition at line 785 of file general.c.

5.286.4.3 signet_field_key_t signet_user_field_keys[256]

Definition at line 405 of file general.c.

5.287 src/providers/dime/dmessage/crypto.h File Reference

```
#include "dime/signet/signet.h"
#include "dime/dmessage/common.h"
```

Data Structures

- struct [object_chunk](#)
- struct [dmime_object_t](#)
- struct [dmime_message_t](#)

Defines

- #define [TRACING_LENGTH_SIZE](#) 2

Typedefs

- typedef struct [object_chunk](#) [dmime_object_chunk_t](#)

Enumerations

- enum [dmime_actor_t](#) { [id_author](#) = 0, [id_origin](#) = 1, [id_destination](#) = 2, [id_recipient](#) = 3 }
- enum [dmime_object_state_t](#) {
[DMIME_OBJECT_STATE_NONE](#) = 0, [DMIME_OBJECT_STATE_CREATION](#), [DMIME_OBJECT_STATE_LOADED_ENVELOPE](#), [DMIME_OBJECT_STATE_LOADED_SIGNETS](#),
[DMIME_OBJECT_STATE_INCOMPLETE_ENVELOPE](#), [DMIME_OBJECT_STATE_INCOMPLETE_METADATA](#), [DMIME_OBJECT_STATE_COMPLETE](#) }
- enum [dmime_message_state_t](#) {
[MESSAGE_STATE_NONE](#) = 0, [MESSAGE_STATE_INCOMPLETE](#), [MESSAGE_STATE_EMPTY](#), [MESSAGE_STATE_ENCODED](#),
[MESSAGE_STATE_CHUNKS_SIGNED](#), [MESSAGE_STATE_ENCRYPTED](#), [MESSAGE_STATE_AUTHOR_SIGNED](#),
[MESSAGE_STATE_COMPLETE](#) }
- enum [dmime_message_chunk_state_t](#) {
[MESSAGE_CHUNK_STATE_NONE](#) = 0, [MESSAGE_CHUNK_STATE_UNKNOWN](#), [MESSAGE_CHUNK_STATE_CREATION](#),
[MESSAGE_CHUNK_STATE_ENCODED](#),
[MESSAGE_CHUNK_STATE_SIGNED](#), [MESSAGE_CHUNK_STATE_ENCRYPTED](#) }

Functions

- struct [__attribute__](#) ((packed))
- char const * [dime_dmsg_actor_to_string](#) ([dmime_actor_t](#) actor)
returns a string from dmime_actor_t.
- int [dime_dmsg_chunks_sig_origin_sign](#) ([dmime_message_t](#) *msg, unsigned char bounce_flags, [dmime_kek_t](#) *kek, [ED25519_KEY](#) *signkey)
signs the encrypted, author signed dmime message with the origin signatures. the origin signature chunks must already exist in order for the signing to occur.
- int [dime_dmsg_kek_in_derive](#) ([dmime_message_t](#) const *msg, [EC_KEY](#) *enckey, [dmime_kek_t](#) *kek)
calculates the key encryption key for a given private encryption key and dmime message, using the ephemeral key chunk in the message

- `dmime_message_t * dime_dmsg_message_binary_deserialize` (unsigned char const *in, size_t insize)
converts a binary message into a dmime message. the message is assumed to be encrypted.
- unsigned char * `dime_dmsg_message_binary_serialize` (dmime_message_t const *msg, unsigned char sections, unsigned char tracing, size_t *outsize)
converts the specified sections of a dmime message to a complete binary form. the message must be at least signed by author.
- int `dime_dmsg_message_decrypt_as_auth` (dmime_object_t *obj, dmime_message_t const *msg, dmime_kek_t *kek)
decrypts, verifies and extracts all the information available to the author from the message.
- int `dime_dmsg_message_decrypt_as_dest` (dmime_object_t *obj, dmime_message_t const *msg, dmime_kek_t *kek)
decrypts, verifies and extracts all the information available to the destination from the message.
- int `dime_dmsg_message_decrypt_as_orig` (dmime_object_t *obj, dmime_message_t const *msg, dmime_kek_t *kek)
decrypts, verifies and extracts all the information available to the origin from the message.
- int `dime_dmsg_message_decrypt_as_recp` (dmime_object_t *obj, dmime_message_t const *msg, dmime_kek_t *kek)
decrypts, verifies and extracts all the information available to the recipient from the message.
- void `dime_dmsg_message_destroy` (dmime_message_t *msg)
destroys dmime_message_t structure.
- `dmime_message_t * dime_dmsg_message_encrypt` (dmime_object_t *object, ED25519_KEY *signkey)
converts a dmime object to a dmime message, fully encrypting and signing the message !!as an author!!
- `dmime_object_t * dime_dmsg_message_envelope_decrypt` (dmime_message_t const *msg, dmime_actor_t actor, dmime_kek_t *kek)
retrieves author name for the following actors: author, origin, recipient.
- `dmime_message_state_t dime_dmsg_message_state_get` (dmime_message_t const *message)
retrieves dmime message state.
- `dmime_object_chunk_t * dime_dmsg_object_chunk_create` (dmime_chunk_type_t type, unsigned char *data, size_t data_size, unsigned char flags)
creates a dmime object chunk with the specified type, data and flags.
- void `dime_dmsg_object_chunklist_destroy` (dmime_object_chunk_t *list)
destroy dmime object chunk list.
- void `dime_dmsg_object_destroy` (dmime_object_t *object)
destroy a dmime_object_t structure.
- int `dime_dmsg_object_dump` (dmime_object_t *object)
dumps the contents of the dmime object.
- `dmime_object_state_t dime_dmsg_object_state_init` (dmime_object_t *object)
takes a dmime object and determines the state it is in.
- const char * `dime_dmsg_object_state_to_string` (dmime_object_state_t state)
returns a string from dmime_object_state_t.

Variables

- [dmime_kek_t](#)
- [dmime_tracing_t](#)
- [dmime_message_chunk_t](#)

5.287.1 Define Documentation

5.287.1.1 #define TRACING_LENGTH_SIZE 2

Definition at line 7 of file crypto.h.

5.287.2 Typedef Documentation

5.287.2.1 typedef struct object_chunk dmime_object_chunk_t

5.287.3 Enumeration Type Documentation

5.287.3.1 enum dmime_actor_t

Enumerator:

id_author
id_origin
id_destination
id_recipient

Definition at line 11 of file crypto.h.

5.287.3.2 enum dmime_message_chunk_state_t

Enumerator:

MESSAGE_CHUNK_STATE_NONE
MESSAGE_CHUNK_STATE_UNKNOWN
MESSAGE_CHUNK_STATE_CREATION
MESSAGE_CHUNK_STATE_ENCODED
MESSAGE_CHUNK_STATE_SIGNED
MESSAGE_CHUNK_STATE_ENCRYPTED

Definition at line 98 of file crypto.h.

5.287.3.3 enum dmime_message_state_t

Enumerator:

MESSAGE_STATE_NONE
MESSAGE_STATE_INCOMPLETE
MESSAGE_STATE_EMPTY
MESSAGE_STATE_ENCODED
MESSAGE_STATE_CHUNKS_SIGNED

MESSAGE_STATE_ENCRYPTED
MESSAGE_STATE_AUTHOR_SIGNED
MESSAGE_STATE_COMPLETE

Definition at line 34 of file crypto.h.

5.287.3.4 enum dmime_object_state_t

Enumerator:

DMIME_OBJECT_STATE_NONE
DMIME_OBJECT_STATE_CREATION
DMIME_OBJECT_STATE_LOADED_ENVELOPE
DMIME_OBJECT_STATE_LOADED_SIGNETS
DMIME_OBJECT_STATE_INCOMPLETE_ENVELOPE
DMIME_OBJECT_STATE_INCOMPLETE_METADATA
DMIME_OBJECT_STATE_COMPLETE

Definition at line 20 of file crypto.h.

5.287.4 Function Documentation

5.287.4.1 struct __attribute__((packed)) [read]

Definition at line 108 of file crypto.h.

References `CHUNK_LENGTH_SIZE`, `data`, and `type()`.

5.287.4.2 char const* dime_dmsg_actor_to_string(dmime_actor_t actor)

returns a string from `dmime_actor_t`.

Parameters:

actor actor value.

Returns:

string containing human readable actor.

Definition at line 6091 of file dmsg.c.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.287.4.3 int dime_dmsg_chunks_sig_origin_sign(dmime_message_t * msg, unsigned char bounce_flags, dmime_kek_t * kek, ED25519_KEY * signkey)

signs the encrypted, author signed dmime message with the origin signatures. the origin signature chunks must already exist in order for the signing to occur.

Parameters:

msg dmime message that will be signed by the origin.

bounce_flags flags indicating bounce signatures that the origin will sign.

kek origin's key encryption key.

signkey origin's private signing key that will be used to sign the message. the public part of this key must be included in the origin signet either as the pok or one of the soks with the message signing flag.

Returns:

0 on success, anything else indicates failure.

Definition at line 6115 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.4 int dime_dmsg_kek_in_derive (dmime_message_t const * msg, EC_KEY * enckey, dmime_kek_t * kek)

calculates the key encryption key for a given private encryption key and dmime message, using the ephemeral key chunk in the message

Parameters:

msg pointer to the dmime message, which has the ephemeral key chunk to be used.

enckey private ec encryption key.

kek pointer to a dmime_kek_t - a key encryption key object that can be used to decrypt the keyslots.

Returns:

returns 0 on success, all other values indicate failure.

Definition at line 6144 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.5 dmime_message_t* dime_dmsg_message_binary_deserialize (unsigned char const * in, size_t insize)

converts a binary message into a dmime message. the message is assumed to be encrypted.

Parameters:

in pointer to the binary message.

insize pointer to the binary size.

Returns:

pointer to a dmime message structure. {dime_dmsg_destroy_message}

Definition at line 6165 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.6 unsigned char* dime_dmsg_message_binary_serialize (dmime_message_t const * msg, unsigned char sections, unsigned char tracing, size_t * outsize)

converts the specified sections of a dmime message to a complete binary form. the message must be at least signed by author.

Parameters:

msg dmime message to be converted.

sections sections to be included.

tracing if set, include tracing, if clear don't include tracing.

outsize stores the output size of the binary. {free}

Definition at line 6187 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.7 int dime_dmsg_message_decrypt_as_auth (dmime_object_t * *obj*, dmime_message_t const * *msg*, dmime_kek_t * *kek*)

decrypts, verifies and extracts all the information available to the author from the message.

Parameters:

obj dmime object into which the information is extracted, it must already contain the ids and signets of all the actors available to the author.

msg dmime message to be decrypted.

kek author's key encryption key.

Returns:

0 on success, all other output values indicate failure.

Definition at line 6216 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.8 int dime_dmsg_message_decrypt_as_dest (dmime_object_t * *obj*, dmime_message_t const * *msg*, dmime_kek_t * *kek*)

decrypts, verifies and extracts all the information available to the destination from the message.

Parameters:

obj dmime object into which the information is extracted, it must already contain the ids and signets of all the actors available to the destination.

msg dmime message to be decrypted.

kek destination's key encryption key.

Returns:

0 on success, all other output values indicate failure.

Definition at line 6243 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.9 int dime_dmsg_message_decrypt_as_orig (dmime_object_t * *obj*, dmime_message_t const * *msg*, dmime_kek_t * *kek*)

decrypts, verifies and extracts all the information available to the origin from the message.

Parameters:

obj dmime object into which the information is extracted, it must already contain the ids and signets of all the actors available to the origin.

msg dmime message to be decrypted.

kek origin's key encryption key.

Returns:

0 on success, all other output values indicate failure.

Definition at line 6266 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.10 int dime_dmsg_message_decrypt_as_recip (dmime_object_t * *obj*, dmime_message_t const * *msg*, dmime_kek_t * *kek*)

decrypts, verifies and extracts all the information available to the recipient from the message.

Parameters:

obj dmime object into which the information is extracted, it must already contain the ids and signets of all the actors available to the recipient.

msg dmime message to be decrypted.

kek recipient's key encryption key.

Returns:

0 on success, all other output values indicate failure.

Definition at line 6289 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.11 void dime_dmsg_message_destroy (dmime_message_t * *msg*)

destroys [dmime_message_t](#) structure.

Parameters:

msg pointer to the dmime message to be destroyed.

Definition at line 6304 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

5.287.4.12 dmime_message_t* dime_dmsg_message_encrypt (dmime_object_t * *object*, ED25519_KEY * *signkey*)

converts a dmime object to a dmime message, fully encrypting and signing the message !!as an author!!

Parameters:

object dmime object which contains all the envelope, metadata, display and attachment information. as well as pointers to signets of author, origin, destination and recipient.

signkey the author's private ed25519 signing key which will be used.

Returns:

a pointer to a fully signed and encrypted dmime message. {dime_dmsg_destroy_message}

Definition at line 6324 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.13 dmime_object_t* dime_dmsg_message_envelope_decrypt (dmime_message_t const * *msg*, dmime_actor_t *actor*, dmime_kek_t * *kek*)

retrieves author name for the following actors: author, origin, recipient.

Parameters:

msg dmime message the author of which is retrieved.
actor who is trying to get the message author.
kek key encryption key for the specified actor.

Returns:

a newly allocated dmime object containing the envelope ids available to the actor. {dime_dmsg_destroy_object}

Definition at line 6346 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.14 dmime_message_state_t dime_dmsg_message_state_get (dmime_message_t const * message)

retrieves dmime message state.

Parameters:

message pointer to a dmime message.

Returns:

dmime_message_state_t corresponding to the current state.

Definition at line 6363 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.15 dmime_object_chunk_t* dime_dmsg_object_chunk_create (dmime_chunk_type_t type, unsigned char * data, size_t data_size, unsigned char flags)

creates a dmime object chunk with the specified type, data and flags.

Parameters:

type chunk type.
data pointer to an array that gets copied into newly allocated memory.
data_size length of data array.
flags specified flags for the object chunk. {dime_dmsg_object_chunklist_destroy}

Definition at line 6383 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.16 void dime_dmsg_object_chunklist_destroy (dmime_object_chunk_t * list)

destroy dmime object chunk list.

Parameters:

list pointer to a dmime object chunk list to be destroyed.

Definition at line 6404 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.17 void dime_dmsg_object_destroy (dmime_object_t * *object*)

destroy a [dmime_object_t](#) structure.

Parameters:

object pointer to dmime object to be destroyed.

Definition at line 6416 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

5.287.4.18 int dime_dmsg_object_dump (dmime_object_t * *object*)

dumps the contents of the dmime object.

Parameters:

object dmime object to be dumped.

Returns:

0 on success, all other values indicate failure.

Definition at line 6430 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.19 dmime_object_state_t dime_dmsg_object_state_init (dmime_object_t * *object*)

takes a dmime object and determines the state it is in.

Parameters:

object dmime object, state of which will be retrieved.

Returns:

the state of dmime object.

Definition at line 6444 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.4.20 const char* dime_dmsg_object_state_to_string (dmime_object_state_t *state*)

returns a string from dmime_object_state_t.

Parameters:

state object state value.

Returns:

string containing human readable dmime object state.

Definition at line 6458 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.287.5 Variable Documentation

5.287.5.1 `dmime_kek_t`

Definition at line 49 of file `crypto.h`.

5.287.5.2 `dmime_message_chunk_t`

Definition at line 115 of file `crypto.h`.

5.287.5.3 `dmime_tracing_t`

Definition at line 93 of file `crypto.h`.

5.288 src/providers/dime/dmessage/dmsg.c File Reference

```
#include "dime/common/misc.h"
#include "dime/dmessage/parse.h"
#include "dime/dmessage/crypto.h"
#include "dime/signet/signet.h"
```

Defines

- #define [CKEY_EMPTY](#) { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, NULL, NULL }

Typedefs

- typedef [dmime_kek_t](#) [dmime_kekset_t](#) [4]
- typedef unsigned char [dmime_ephemeral_payload_t](#) [EC_PUBKEY_SIZE]
- typedef unsigned char [dmime_signature_payload_t](#) [ED25519_SIG_SIZE]
- typedef unsigned char * [dmime_encrypted_payload_t](#)

Functions

- struct [__attribute__](#) ((packed))
- char const * [dime_dmsg_actor_to_string](#) ([dmime_actor_t](#) actor)
returns a string from dmime_actor_t.
- int [dime_dmsg_chunks_sig_origin_sign](#) ([dmime_message_t](#) *msg, unsigned char bounce_flags, [dmime_kek_t](#) *kek, [ED25519_KEY](#) *signkey)
signs the encrypted, author signed dmime message with the origin signatures. the origin signature chunks must already exist in order for the signing to occur.
- int [dime_dmsg_kek_in_derive](#) ([dmime_message_t](#) const *msg, [EC_KEY](#) *enckey, [dmime_kek_t](#) *kek)
calculates the key encryption key for a given private encryption key and dmime message, using the ephemeral key chunk in the message
- [dmime_message_t](#) * [dime_dmsg_message_binary_deserialize](#) (unsigned char const *in, size_t insize)
converts a binary message into a dmime message. the message is assumed to be encrypted.
- unsigned char * [dime_dmsg_message_binary_serialize](#) ([dmime_message_t](#) const *msg, unsigned char sections, unsigned char tracing, size_t *outsize)
converts the specified sections of a dmime message to a complete binary form. the message must be at least signed by author.
- int [dime_dmsg_message_decrypt_as_auth](#) ([dmime_object_t](#) *obj, [dmime_message_t](#) const *msg, [dmime_kek_t](#) *kek)
decrypts, verifies and extracts all the information available to the author from the message.
- int [dime_dmsg_message_decrypt_as_dest](#) ([dmime_object_t](#) *obj, [dmime_message_t](#) const *msg, [dmime_kek_t](#) *kek)
decrypts, verifies and extracts all the information available to the destination from the message.
- int [dime_dmsg_message_decrypt_as_orig](#) ([dmime_object_t](#) *obj, [dmime_message_t](#) const *msg, [dmime_kek_t](#) *kek)
decrypts, verifies and extracts all the information available to the origin from the message.
- int [dime_dmsg_message_decrypt_as_recip](#) ([dmime_object_t](#) *obj, [dmime_message_t](#) const *msg, [dmime_kek_t](#) *kek)
decrypts, verifies and extracts all the information available to the recipient from the message.

- void `dime_dmsg_message_destroy` (`dmime_message_t` *msg)
destroys `dmime_message_t` structure.
- `dmime_message_t` * `dime_dmsg_message_encrypt` (`dmime_object_t` *object, `ED25519_KEY` *signkey)
converts a `dmime` object to a `dmime` message, fully encrypting and signing the message !!as an author!!
- `dmime_object_t` * `dime_dmsg_message_envelope_decrypt` (`dmime_message_t` const *msg, `dmime_actor_t` actor, `dmime_kek_t` *kek)
retrieves author name for the following actors: author, origin, recipient.
- `dmime_message_state_t` `dime_dmsg_message_state_get` (`dmime_message_t` const *message)
retrieves `dmime` message state.
- `dmime_object_chunk_t` * `dime_dmsg_object_chunk_create` (`dmime_chunk_type_t` type, unsigned char *data, size_t data_size, unsigned char flags)
creates a `dmime` object chunk with the specified type, data and flags.
- void `dime_dmsg_object_chunklist_destroy` (`dmime_object_chunk_t` *list)
destroy `dmime` object chunk list.
- void `dime_dmsg_object_destroy` (`dmime_object_t` *object)
destroy a `dmime_object_t` structure.
- int `dime_dmsg_object_dump` (`dmime_object_t` *object)
dumps the contents of the `dmime` object.
- `dmime_object_state_t` `dime_dmsg_object_state_init` (`dmime_object_t` *object)
takes a `dmime` object and determines the state it is in.
- const char * `dime_dmsg_object_state_to_string` (`dmime_object_state_t` state)
returns a string from `dmime_object_state_t`.

Variables

- `dmime_standard_payload_t`
- `dmime_keyslot_t`
- `dmime_chunk_key_t` `dmime_chunk_keys` [`DMIME_CHUNK_TYPE_MAX`]

5.288.1 Define Documentation

5.288.1.1 #define CKEY_EMPTY { 0, 0, 0, 0, 0, 0, 0, 0, 0, NULL, NULL }

Definition at line 6488 of file `dmsg.c`.

5.288.2 Typedef Documentation

5.288.2.1 typedef unsigned char* dmime_encrypted_payload_t

Definition at line 24 of file `dmsg.c`.

5.288.2.2 typedef unsigned char dmime_ephemeral_payload_t[EC_PUBKEY_SIZE]

Definition at line 9 of file dmsg.c.

5.288.2.3 typedef dmime_kek_t dmime_kekset_t[4]

Definition at line 6 of file dmsg.c.

5.288.2.4 typedef unsigned char dmime_signature_payload_t[ED25519_SIG_SIZE]

Definition at line 21 of file dmsg.c.

5.288.3 Function Documentation**5.288.3.1 struct __attribute__((packed)) [read]**

Definition at line 27 of file dmsg.c.

References AES_256_KEY_SIZE.

5.288.3.2 char const* dime_dmsg_actor_to_string(dmime_actor_t actor)

returns a string from dmime_actor_t.

Parameters:

actor actor value.

Returns:

string containing human readable actor.

Definition at line 6091 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.3 int dime_dmsg_chunks_sig_origin_sign(dmime_message_t *msg, unsigned char bounce_flags, dmime_kek_t *kek, ED25519_KEY *signkey)

signs the encrypted, author signed dmime message with the origin signatures. the origin signature chunks must already exist in order for the signing to occur.

Parameters:

msg dmime message that will be signed by the origin.

bounce_flags flags indicating bounce signatures that the origin will sign.

kek origin's key encryption key.

signkey origin's private signing key that will be used to sign the message. the public part of this key must be included in the origin signet either as the pok or one of the soks with the message signing flag.

Returns:

0 on success, anything else indicates failure.

Definition at line 6115 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.4 int dime_dmsg_kek_in_derive (dmime_message_t const * *msg*, EC_KEY * *enckey*, dmime_kek_t * *kek*)

calculates the key encryption key for a given private encryption key and dmime message, using the ephemeral key chunk in the message

Parameters:

msg pointer to the dmime message, which has the ephemeral key chunk to be used.

enckey private ec encryption key.

kek pointer to a dmime_kek_t - a key encryption key object that can be used to decrypt the keyslots.

Returns:

returns 0 on success, all other values indicate failure.

Definition at line 6144 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.5 dmime_message_t* dime_dmsg_message_binary_deserialize (unsigned char const * *in*, size_t *insize*)

converts a binary message into a dmime message. the message is assumed to be encrypted.

Parameters:

in pointer to the binary message.

insize pointer to the binary size.

Returns:

pointer to a dmime message structure. {dime_dmsg_destroy_message}

Definition at line 6165 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.6 unsigned char* dime_dmsg_message_binary_serialize (dmime_message_t const * *msg*, unsigned char *sections*, unsigned char *tracing*, size_t * *outsize*)

converts the specified sections of a dmime message to a complete binary form. the message must be at least signed by author.

Parameters:

msg dmime message to be converted.

sections sections to be included.

tracing if set, include tracing, if clear don't include tracing.

outsize stores the output size of the binary. {free}

Definition at line 6187 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.7 int dime_dmsg_message_decrypt_as_auth (dmime_object_t * *obj*, dmime_message_t const * *msg*, dmime_kek_t * *kek*)

decrypts, verifies and extracts all the information available to the author from the message.

Parameters:

obj dmime object into which the information is extracted, it must already contain the ids and signets of all the actors available to the author.

msg dmime message to be decrypted.

kek author's key encryption key.

Returns:

0 on success, all other output values indicate failure.

Definition at line 6216 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.8 int dime_dmsg_message_decrypt_as_dest (dmime_object_t * *obj*, dmime_message_t const * *msg*, dmime_kek_t * *kek*)

decrypts, verifies and extracts all the information available to the destination from the message.

Parameters:

obj dmime object into which the information is extracted, it must already contain the ids and signets of all the actors available to the destination.

msg dmime message to be decrypted.

kek destination's key encryption key.

Returns:

0 on success, all other output values indicate failure.

Definition at line 6243 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.9 int dime_dmsg_message_decrypt_as_orig (dmime_object_t * *obj*, dmime_message_t const * *msg*, dmime_kek_t * *kek*)

decrypts, verifies and extracts all the information available to the origin from the message.

Parameters:

obj dmime object into which the information is extracted, it must already contain the ids and signets of all the actors available to the origin.

msg dmime message to be decrypted.

kek origin's key encryption key.

Returns:

0 on success, all other output values indicate failure.

Definition at line 6266 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.10 int dime_dmsg_message_decrypt_as_recip (dmime_object_t * *obj*, dmime_message_t const * *msg*, dmime_kek_t * *kek*)

decrypts, verifies and extracts all the information available to the recipient from the message.

Parameters:

obj dmime object into which the information is extracted, it must already contain the ids and signets of all the actors available to the recipient.

msg dmime message to be decrypted.

kek recipient's key encryption key.

Returns:

0 on success, all other output values indicate failure.

Definition at line 6289 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.11 void dime_dmsg_message_destroy (dmime_message_t * *msg*)

destroys [dmime_message_t](#) structure.

Parameters:

msg pointer to the dmime message to be destroyed.

Definition at line 6304 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

5.288.3.12 dmime_message_t* dime_dmsg_message_encrypt (dmime_object_t * *object*, ED25519_KEY * *signkey*)

converts a dmime object to a dmime message, fully encrypting and signing the message !!as an author!!

Parameters:

object dmime object which contains all the envelope, metadata, display and attachment information. as well as pointers to signets of author, origin, destination and recipient.

signkey the author's private ed25519 signing key which will be used.

Returns:

a pointer to a fully signed and encrypted dmime message. {dime_dmsg_destroy_message}

Definition at line 6324 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.13 dmime_object_t* dime_dmsg_message_envelope_decrypt (dmime_message_t const * *msg*, dmime_actor_t *actor*, dmime_kek_t * *kek*)

retrieves author name for the following actors: author, origin, recipient.

Parameters:

msg dmime message the author of which is retrieved.

actor who is trying to get the message author.

kek key encryption key for the specified actor.

Returns:

a newly allocated dmime object containing the envelope ids available to the actor. {dime_dmsg_destroy_object}

Definition at line 6346 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.14 dmime_message_state_t dime_dmsg_message_state_get (dmime_message_t const * *message*)

retrieves dmime message state.

Parameters:

message pointer to a dmime message.

Returns:

dmime_message_state_t corresponding to the current state.

Definition at line 6363 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.15 dmime_object_chunk_t* dime_dmsg_object_chunk_create (dmime_chunk_type_t *type*, unsigned char * *data*, size_t *data_size*, unsigned char *flags*)

creates a dmime object chunk with the specified type, data and flags.

Parameters:

type chunk type.

data pointer to an array that gets copied into newly allocated memory.

data_size length of data array.

flags specified flags for the object chunk. {dime_dmsg_object_chunklist_destroy}

Definition at line 6383 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.16 void dime_dmsg_object_chunklist_destroy (dmime_object_chunk_t * *list*)

destroy dmime object chunk list.

Parameters:

list pointer to a dmime object chunk list to be destroyed.

Definition at line 6404 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.17 void dime_dmsg_object_destroy (dmime_object_t * *object*)

destroy a [dmime_object_t](#) structure.

Parameters:

object pointer to dmime object to be destroyed.

Definition at line 6416 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

5.288.3.18 int dime_dmsg_object_dump (dmime_object_t * *object*)

dumps the contents of the dmime object.

Parameters:

object dmime object to be dumped.

Returns:

0 on success, all other values indicate failure.

Definition at line 6430 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.19 dmime_object_state_t dime_dmsg_object_state_init (dmime_object_t * *object*)

takes a dmime object and determines the state it is in.

Parameters:

object dmime object, state of which will be retrieved.

Returns:

the state of dmime object.

Definition at line 6444 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.3.20 const char* dime_dmsg_object_state_to_string (dmime_object_state_t *state*)

returns a string from dmime_object_state_t.

Parameters:

state object state value.

Returns:

string containing human readable dmime object state.

Definition at line 6458 of file dmsg.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.288.4 Variable Documentation

5.288.4.1 dmime_chunk_key_t dmime_chunk_keys[DMIME_CHUNK_TYPE_MAX]

Definition at line 6491 of file dmsg.c.

5.288.4.2 dmime_keyslot_t

Definition at line 31 of file dmsg.c.

5.288.4.3 `dmime_standard_payload_t`

Definition at line 18 of file `dmsg.c`.

5.289 src/providers/dime/dmessage/parse.h File Reference

```
#include "dime/dmessage/common.h"
```

Data Structures

- struct [dmime_header_key_t](#)
- struct [dmime_envelope_object_t](#)

Enumerations

- enum [dmime_header_type_t](#) {
[HEADER_TYPE_DATE](#) = 0, [HEADER_TYPE_TO](#), [HEADER_TYPE_CC](#), [HEADER_TYPE_FROM](#),
[HEADER_TYPE_ORGANIZATION](#), [HEADER_TYPE_SUBJECT](#), [HEADER_TYPE_NONE](#) }

Functions

- void [dime_prsr_envelope_destroy](#) ([dmime_envelope_object_t](#) *obj)
Destroys a [dmime_envelop_object_t](#) structure.
- sds [dime_prsr_envelope_format](#) (sds user_id, sds org_id, char const *user_fp, char const *org_fp, [dmime_chunk_type_t](#) type)
Formats the passed in envelope data into a string to be passed to an envelope chunk.
- [dmime_envelope_object_t](#) * [dime_prsr_envelope_parse](#) (char const *in, size_t insize, [dmime_chunk_type_t](#) type)
Parses a binary buffer from a dmime message into a dmime origin object.
- [dmime_common_headers_t](#) * [dime_prsr_headers_create](#) (void)
Allocates memory for an empty [dmime_common_headers_t](#) type.
- void [dime_prsr_headers_destroy](#) ([dmime_common_headers_t](#) *obj)
Destroys a [dmime_common_headers_t](#) structure.
- unsigned char * [dime_prsr_headers_format](#) ([dmime_common_headers_t](#) *obj, size_t *outsize)
Formats the [dmime_common_headers_t](#) into a single array for the common headers chunk.
- [dmime_common_headers_t](#) * [dime_prsr_headers_parse](#) (unsigned char *in, size_t insize)
Parses the passed array of bytes into [dmime_common_headers_t](#).

Variables

- [dmime_header_key_t](#) [dmime_header_keys](#) [[DMIME_NUM_COMMON_HEADERS](#)]

5.289.1 Enumeration Type Documentation

5.289.1.1 enum dmime_header_type_t

Enumerator:

[HEADER_TYPE_DATE](#)
[HEADER_TYPE_TO](#)

HEADER_TYPE_CC
HEADER_TYPE_FROM
HEADER_TYPE_ORGANIZATION
HEADER_TYPE_SUBJECT
HEADER_TYPE_NONE

Definition at line 21 of file parse.h.

5.289.2 Function Documentation

5.289.2.1 void dime_prsr_envelope_destroy (dmime_envelope_object_t * *obj*)

Destroys a dmime_envelop_object_t structure.

Parameters:

obj Pointer to the object to be destroyed.

Definition at line 751 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.289.2.2 sds dime_prsr_envelope_format (sds *user_id*, sds *org_id*, char const * *user_fp*, char const * *org_fp*, dmime_chunk_type_t *type*)

Formats the passed in envelope data into a string to be passed to an envelope chunk.

Parameters:

user_id Stringer containing the user email address.

org_id Stringer containing the organizational TLD.

user_fp Cstring containing cryptographic user signet.

org_fp Cstring containing organizational signet fingerprint.

type The type of envelope chunk.

Returns:

Buffer with formatted envelope data. {st_free}

Definition at line 776 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.289.2.3 dmime_envelope_object_t* dime_prsr_envelope_parse (char const * *in*, size_t *insize*, dmime_chunk_type_t *type*)

Parses a binary buffer from a dmime message into a dmime origin object.

Parameters:

in Binary origin array.

insize Size of input array.

type Type of the chunk.

Returns:

Pointer to a parsed dmime object or NULL on error. {dime_prsr_envelope_destroy}

Definition at line 800 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.289.2.4 dmime_common_headers_t* dime_prsr_headers_create (void)

Allocates memory for an empty [dmime_common_headers_t](#) type.

Returns:

[dmime_common_headers_t](#) structure. {dime_prsr_headers_destroy}

Definition at line 816 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.289.2.5 void dime_prsr_headers_destroy (dmime_common_headers_t * obj)

Destroys a [dmime_common_headers_t](#) structure.

Parameters:

obj Headers to be destroyed.

Definition at line 828 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

5.289.2.6 unsigned char* dime_prsr_headers_format (dmime_common_headers_t * obj, size_t * outsize)

Formats the [dmime_common_headers_t](#) into a single array for the common headers chunk.

Parameters:

obj The headers to be formatted.

outsize Stores the size of the output array.

Returns:

Returns the array of ASCII characters (not terminated by ") as pointer to unsigned char. {free}

Definition at line 846 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.289.2.7 dmime_common_headers_t* dime_prsr_headers_parse (unsigned char * in, size_t insize)

Parses the passed array of bytes into [dmime_common_headers_t](#).

Parameters:

in Input buffer.

insize Input buffer size.

Returns:

A [dmime_common_headers_t](#) array of sds strings containing parsed header info. {dime_prsr_headers_destroy}

Definition at line 865 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.289.3 Variable Documentation

5.289.3.1 `dmime_header_key_t dmime_header_keys[DMIME_NUM_COMMON_HEADERS]`

Definition at line 732 of file parser.c.

5.290 src/providers/dime/dmessage/parser.c File Reference

```
#include "dime/common/misc.h"
#include "dime/dmessage/parse.h"
```

Functions

- void [dime_prsr_envelope_destroy](#) ([dmime_envelope_object_t](#) *obj)
Destroys a [dmime_envelop_object_t](#) structure.
- [sds](#) [dime_prsr_envelope_format](#) ([sds](#) user_id, [sds](#) org_id, char const *user_fp, char const *org_fp, [dmime_chunk_type_t](#) type)
Formats the passed in envelope data into a string to be passed to an envelope chunk.
- [dmime_envelope_object_t](#) * [dime_prsr_envelope_parse](#) (char const *in, size_t insize, [dmime_chunk_type_t](#) type)
Parses a binary buffer from a dmime message into a dmime origin object.
- [dmime_common_headers_t](#) * [dime_prsr_headers_create](#) (void)
Allocates memory for an empty [dmime_common_headers_t](#) type.
- void [dime_prsr_headers_destroy](#) ([dmime_common_headers_t](#) *obj)
Destroys a [dmime_common_headers_t](#) structure.
- unsigned char * [dime_prsr_headers_format](#) ([dmime_common_headers_t](#) *obj, size_t *outsize)
Formats the [dmime_common_headers_t](#) into a single array for the common headers chunk.
- [dmime_common_headers_t](#) * [dime_prsr_headers_parse](#) (unsigned char *in, size_t insize)
Parses the passed array of bytes into [dmime_common_headers_t](#).

Variables

- [dmime_header_key_t](#) [dmime_header_keys](#) [DMIME_NUM_COMMON_HEADERS]

5.290.1 Function Documentation

5.290.1.1 void [dime_prsr_envelope_destroy](#) ([dmime_envelope_object_t](#) *obj)

Destroys a [dmime_envelop_object_t](#) structure.

Parameters:

obj Pointer to the object to be destroyed.

Definition at line 751 of file parser.c.

References [PUBLIC_FUNCTION_IMPLEMENT](#).

5.290.1.2 [sds](#) [dime_prsr_envelope_format](#) ([sds](#) user_id, [sds](#) org_id, char const *user_fp, char const *org_fp, [dmime_chunk_type_t](#) type)

Formats the passed in envelope data into a string to be passed to an envelope chunk.

Parameters:

user_id Stringer containing the user email address.
org_id Stringer containing the organizational TLD.
user_fp Cstring containing cryptographic user signet.
org_fp Cstring containing organizational signet fingerprint.
type The type of envelope chunk.

Returns:

Buffer with formatted envelope data. {st_free}

Definition at line 776 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.290.1.3 dmime_envelope_object_t* dime_prsr_envelope_parse (char const * in, size_t insize, dmime_chunk_type_t type)

Parses a binary buffer from a dmime message into a dmime origin object.

Parameters:

in Binary origin array.
insize Size of input array.
type Type of the chunk.

Returns:

Pointer to a parsed dmime object or NULL on error. {dime_prsr_envelope_destroy}

Definition at line 800 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.290.1.4 dmime_common_headers_t* dime_prsr_headers_create (void)

Allocates memory for an empty [dmime_common_headers_t](#) type.

Returns:

[dmime_common_headers_t](#) structure. {dime_prsr_headers_destroy}

Definition at line 816 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.290.1.5 void dime_prsr_headers_destroy (dmime_common_headers_t * obj)

Destroys a [dmime_common_headers_t](#) structure.

Parameters:

obj Headers to be destroyed.

Definition at line 828 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

5.290.1.6 unsigned char* dime_prsr_headers_format (dmime_common_headers_t * *obj*, size_t * *outsize*)

Formats the [dmime_common_headers_t](#) into a single array for the common headers chunk.

Parameters:

obj The headers to be formatted.

outsize Stores the size of the output array.

Returns:

Returns the array of ASCII characters (not terminated by `”`) as pointer to unsigned char. {free}

Definition at line 846 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.290.1.7 dmime_common_headers_t* dime_prsr_headers_parse (unsigned char * *in*, size_t *insize*)

Parses the passed array of bytes into [dmime_common_headers_t](#).

Parameters:

in Input buffer.

insize Input buffer size.

Returns:

A [dmime_common_headers_t](#) array of sds strings containing parsed header info. {dime_prsr_headers_destroy}

Definition at line 865 of file parser.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.290.2 Variable Documentation**5.290.2.1 dmime_header_key_t dmime_header_keys[DMIME_NUM_COMMON_HEADERS]****Initial value:**

```
{
    {1, "Date: ", 6},
    {1, "To: ", 4},
    {0, "CC: ", 4},
    {1, "From: ", 6},
    {0, "Organization: ", 14},
    {1, "Subject: ", 9},
    {0, NULL, 0}
}
```

Definition at line 732 of file parser.c.

5.291 src/providers/dime/dmtp/commands.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "dime/sds/sds.h"
#include "dime/dmtp/commands.h"
#include "dime/common/error.h"
```

Defines

- #define [DMTP_IGNORE_ARG](#) { NULL, 0, DMTP_ARG_PLAIN, 0 }
- #define [DMTP_EMPTY_ARG](#) { NULL, 0, DMTP_ARG_NONE, 0 }

Functions

- [dmtp_command_t * dime_dmtp_command_create](#) (dmtp_command_type_t type)
Create a dmtp command structure.
- void [dime_dmtp_command_destroy](#) (dmtp_command_t *command)
Destroy a dmtp command structure.
- [sds dime_dmtp_command_format](#) (dmtp_command_t *command)
Formats the specified dmtp command into a sds string.
- [dmtp_parse_state_t dime_dmtp_command_parse](#) (sds comm_line, dmtp_command_t **comm_struct)
Parse the provided sds string into a dmtp command structure.
- [sds dime_dmtp_command_starttls](#) (sds host, dmtp_mode_t mode)
Create a string containing dmtp STARTTLS command with specified arguments.
- [sds dime_dmtp_command_helo](#) (sds host)
Create a string containing dmtp HELO command with specified arguments.
- [sds dime_dmtp_command_ehlo](#) (sds host)
Create a string containing dmtp EHLO command with specified arguments.
- [sds dime_dmtp_command_mode](#) ()
Create a string containing dmtp MODE command with specified arguments.
- [sds dime_dmtp_command_rset](#) ()
Create a string containing dmtp RSET command with specified arguments.
- [sds dime_dmtp_command_noop](#) (sds optarg1, sds optarg2, sds optarg3)
Create a string containing dmtp NOOP command with specified arguments.
- [sds dime_dmtp_command_help](#) ()
Create a string containing dmtp HELP command with specified arguments.
- [sds dime_dmtp_command_quit](#) ()

Create a string containing dmtp *QUIT* command with specified arguments.

- `sds dime_dmtp_command_mail` (`sds` from, `sds` fingerprint)
Create a string containing dmtp *MAIL* command with specified arguments.
- `sds dime_dmtp_command_rcpt` (`sds` to, `sds` fingerprint)
Create a string containing dmtp *RCPT* command with specified arguments.
- `sds dime_dmtp_command_data` ()
Create a string containing dmtp *DATA* command with specified arguments.
- `sds dime_dmtp_command_sgt_user` (`sds` address, `sds` fingerprint)
Create a string containing dmtp *SGNT* command with specified arguments.
- `sds dime_dmtp_command_sgt_domain` (`sds` domain, `sds` fingerprint)
Create a string containing dmtp *SGNT* command with specified arguments.
- `sds dime_dmtp_command_hist` (`sds` address, `sds` start, `sds` stop)
Create a string containing dmtp *HIST* command with specified arguments.
- `sds dime_dmtp_command_vrfy_user` (`sds` address, `sds` fingerprint)
Create a string containing dmtp *VRFY* command with specified arguments.
- `sds dime_dmtp_command_vrfy_domain` (`sds` domain, `sds` fingerprint)
Create a string containing dmtp *VRFY* command with specified arguments.

Variables

- `dmtp_command_key_t dmtp_command_list` [DMTP_COMMANDS_NUM]

5.291.1 Define Documentation

5.291.1.1 `#define DMTP_EMPTY_ARG { NULL, 0, DMTP_ARG_NONE, 0 }`

Definition at line 9 of file commands.c.

5.291.1.2 `#define DMTP_IGNORE_ARG { NULL, 0, DMTP_ARG_PLAIN, 0 }`

Definition at line 8 of file commands.c.

5.291.2 Function Documentation

5.291.2.1 `dmtp_command_t* dime_dmtp_command_create(dmtp_command_type_t type)`

Create a dmtp command structure.

Parameters:

type dmtp command enum.

Returns:

A newly created dmtp command structure with specified type and NULL args. `dime_dmtp_command_destroy`

Definition at line 1086 of file commands.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.291.2.2 sds dime_dmtplib_command_data (void)

Create a string containing dmtplib DATA command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1671 of file commands.c.

References DMTP_DATA, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.291.2.3 void dime_dmtplib_command_destroy (dmtplib_command_t * *command*)

Destroy a dmtplib command structure.

Parameters:

command dmtplib command structure to be destroyed.

Definition at line 1099 of file commands.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.291.2.4 sds dime_dmtplib_command_ehlo (sds *host*)

Create a string containing dmtplib EHLO command with specified arguments.

Parameters:

host Name of the domain that is being connected to.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1283 of file commands.c.

References dmtplib_command_t::args, DMTP_EHLO, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.5 sds dime_dmtplib_command_format (dmtplib_command_t * *command*)

Formats the specified dmtplib command into a sds string.

Parameters:

command Command to be formatted.

Returns:

sds string containing formatted command. sdsfree

Definition at line 1116 of file commands.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.291.2.6 sds dime_dmtp_command_helo (sds *host*)

Create a string containing dmtp HELO command with specified arguments.

Parameters:

host Name of the domain that is being connected to.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1232 of file commands.c.

References dmtp_command_t::args, DMTP_HELO, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.7 sds dime_dmtp_command_help (void)

Create a string containing dmtp HELP command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1468 of file commands.c.

References DMTP_HELP, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.291.2.8 sds dime_dmtp_command_hist (sds *address*, sds *start*, sds *stop*)

Create a string containing dmtp HIST command with specified arguments.

Parameters:

address sds string containing user mail address.

start sds string containing optional starting user signet fingerprint. NULL if unwanted.

stop sds string containing optional ending user signet fingerprint. NULL if unwanted.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1832 of file commands.c.

References dmtp_command_t::args, DMTP_HIST, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.9 sds dime_dmtp_command_mail (sds *from*, sds *fingerprint*)

Create a string containing dmtp MAIL command with specified arguments.

Parameters:

from sds string containing origin domain.

fingerprint sds string containing origin signet full fingerprint.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1547 of file commands.c.

References dmtplib_command_t::args, DMTP_MAIL, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.10 sds dime_dmtplib_command_mode (void)

Create a string containing dmtplib MODE command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1332 of file commands.c.

References DMTP_MODE, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.291.2.11 sds dime_dmtplib_command_noop (sds optarg1, sds optarg2, sds optarg3)

Create a string containing dmtplib NOOP command with specified arguments.

Parameters:

optarg1 Optional allowed argument 1. NULL if unwanted.

optarg2 Optional allowed argument 2. NULL if unwanted.

optarg3 Optional allowed argument 3. NULL if unwanted.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1412 of file commands.c.

References dmtplib_command_t::args, DMTP_NOOP, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.12 dmtplib_parse_state_t dime_dmtplib_command_parse (sds comm_line, dmtplib_command_t ** comm_struct)

Parse the provided sds string into a dmtplib command structure.

Parameters:

command sds dmtplib command string.

Returns:

A valid dmtplib command structure on success, NULL on failure. dime_dmtplib_command_destroy

Definition at line 1133 of file commands.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.291.2.13 sds dime_dmtp_command_quit (void)

Create a string containing dmtp QUIT command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1505 of file commands.c.

References DMTP_QUIT, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.291.2.14 sds dime_dmtp_command_rcpt (sds to, sds fingerprint)

Create a string containing dmtp RCPT command with specified arguments.

Parameters:

to sds string containg destination domain.

fingerprint sds string containing destination signet full fingerprint.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1611 of file commands.c.

References dmtp_command_t::args, DMTP_RCPT, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.15 sds dime_dmtp_command_rset (void)

Create a string containing dmtp RSET command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1369 of file commands.c.

References DMTP_RSET, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.291.2.16 sds dime_dmtp_command_sgnt_domain (sds domain, sds fingerprint)

Create a string containing dmtp SGNT command with specified arguments.

Parameters:

domain sds string containing domain name.

fingerprint sds string containing optional organizational signet fingerprint. NULL if unwanted.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1771 of file commands.c.

References dmtp_command_t::args, DMTP_SGNT, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.17 `sds dime_dmtplib_command_sgnt_user` (*sds address*, *sds fingerprint*)

Create a string containing dmtplib SGNT command with specified arguments.

Parameters:

address sds string containing mail address of the user.

fingerprint sds string containing optional user signet fingerprint. NULL if unwanted.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1712 of file commands.c.

References dmtplib_command_t::args, DMTP_SGNT, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.18 `sds dime_dmtplib_command_starttls` (*sds host*, *dmtplib_mode_t mode*)

Create a string containing dmtplib STARTTLS command with specified arguments.

Parameters:

host Name of the domain that is being connected to.

mode optional mode parameter (DMTP or SMTP). DMTP_MODE_NONE to not specify mode.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1154 of file commands.c.

References dmtplib_command_t::args, DMTP_MODE_DMTP, DMTP_MODE_NONE, DMTP_MODE_SMTP, DMTP_STARTTLS, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, sdsdup(), and sdsnewlen().

5.291.2.19 `sds dime_dmtplib_command_vrfy_domain` (*sds domain*, *sds fingerprint*)

Create a string containing dmtplib VRFY command with specified arguments.

Parameters:

domain sds string containing domain name.

start sds string containing organizational signet fingerprint.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1961 of file commands.c.

References dmtplib_command_t::args, DMTP_VRFY, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.2.20 `sds dime_dmtplib_command_vrfy_user` (*sds address*, *sds fingerprint*)

Create a string containing dmtplib VRFY command with specified arguments.

Parameters:

address sds string containing user mail address.

start sds string containing user signet fingerprint.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1897 of file commands.c.

References dmtp_command_t::args, DMTP_VRFY, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.291.3 Variable Documentation

5.291.3.1 dmtp_command_key_t dmtp_command_list[DMTP_COMMANDS_NUM]

Definition at line 2013 of file commands.c.

5.292 src/servers/dmtp/commands.c File Reference

```
#include "magma.h"
#include "commands.h"
```

Functions

- [int_t dmtp_compare](#) (const void *compare, const void *[command](#))
- void [dmtp_sort](#) (void)
Sort the DMTP command table to be ready for binary searches.
- void [dmtp_requeue](#) ([connection_t](#) *con)
commands.c
- void [dmtp_process](#) ([connection_t](#) *con)
The main entry point in the DMTP server for processing commands issued by clients.

5.292.1 Function Documentation

5.292.1.1 int_t dmtp_compare (const void * *compare*, const void * *command*)

Definition at line 11 of file `commands.c`.

References `command_t`, `PLACER`, `st_cmp_ci_eq()`, and `st_cmp_ci_starts()`.

Referenced by `dmtp_process()`, and `dmtp_sort()`.

5.292.1.2 void dmtp_process (connection_t * *con*)

The main entry point in the DMTP server for processing commands issued by clients.

Parameters:

con a pointer to the connection object of the client issuing the DMTP command.

Returns:

This function returns no value.

Definition at line 48 of file `commands.c`.

References `command`, `command_t`, `con_read_line()`, `dmtp_commands`, `dmtp_compare()`, `dmtp_data()`, `dmtp_invalid()`, `dmtp_process()`, `dmtp_quit()`, `dmtp_requeue()`, `enqueue()`, `pl_char_get()`, `pl_empty()`, `pl_length_get()`, and `requeue()`.

Referenced by `dmtp_process()`, and `dmtp_requeue()`.

5.292.1.3 void dmtp_requeue (connection_t * *con*)

`commands.c`

Definition at line 31 of file `commands.c`.

References `con_status()`, `dmtp_process()`, `dmtp_quit()`, `enqueue()`, and `status`.

Referenced by `dmtp_data()`, `dmtp_init()`, and `dmtp_process()`.

5.292.1.4 void dmtp_sort (void)

Sort the DMTP command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 26 of file commands.c.

References `command_t`, `dmtp_commands`, and `dmtp_compare()`.

Referenced by `protocol_init()`.

5.293 src/servers/imap/commands.c File Reference

```
#include "magma.h"
#include "commands.h"
```

Functions

- `int_t imap_compare` (const void *compare, const void *command)
Compare the names of two commands.
- void `imap_sort` (void)
Sort the IMAP command table to be ready for binary searches.
- void `imap_requeue` (connection_t *con)
Requeue an imap connection for processing, or log it out if there was an error or excess of protocol violations.
- void `imap_process` (connection_t *con)
Perform client command processing on an established imap session.

5.293.1 Function Documentation

5.293.1.1 int_t imap_compare (const void *compare, const void *command)

Compare the names of two commands. commands.c

Note:

This is an internal function used to sort imap commands and search for them.

Parameters:

compare a pointer to the first command to be compared.
command a pointer to the second command to be compared.

Returns:

-1 if compare < command, 1 if command < compare, or 0 if the two commands are equal.

Definition at line 18 of file commands.c.

References `command_t`, `PLACER`, `st_cmp_ci_eq()`, and `st_cmp_ci_starts()`.

Referenced by `imap_process()`, and `imap_sort()`.

5.293.1.2 void imap_process (connection_t *con)

Perform client command processing on an established imap session.

Note:

This function will read the next line of user input, parse the command, and then attempt to execute it with the appropriate handler.

Parameters:

con a pointer to the connection object underlying the imap session.

Returns:

This function returns no value.

Definition at line 61 of file commands.c.

References `command`, `command_t`, `con_print()`, `con_read_line()`, `con_write_bl()`, `enqueue()`, `imap_command_parser()`, `imap_commands`, `imap_compare()`, `imap_invalid()`, `imap_logout()`, `imap_process()`, `imap_requeue()`, `pl_empty()`, `requeue()`, `st_char_get()`, `st_length_get()`, and `st_length_int()`.

Referenced by `imap_process()`, and `imap_requeue()`.

5.293.1.3 void imap_requeue (connection_t * con)

Requeue an imap connection for processing, or log it out if there was an error or excess of protocol violations.

Parameters:

con a pointer to the connection object of the imap session.

Returns:

This function returns no value.

Definition at line 43 of file commands.c.

References `con_status()`, `enqueue()`, `imap_logout()`, `imap_process()`, and `status`.

Referenced by `imap_init()`, and `imap_process()`.

5.293.1.4 void imap_sort (void)

Sort the IMAP command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 33 of file commands.c.

References `command_t`, `imap_commands`, and `imap_compare()`.

Referenced by `protocol_init()`.

5.294 src/servers/molten/commands.c File Reference

```
#include "magma.h"
#include "commands.h"
```

Functions

- [int_t molten_compare](#) (const void *compare, const void *command)
commands.c
- void [molten_sort](#) (void)
Sort the Molten command table to be ready for binary searches.
- void [molten_parse](#) ([connection_t](#) *con)

5.294.1 Function Documentation

5.294.1.1 int_t molten_compare (const void *compare, const void *command)

commands.c

Definition at line 12 of file commands.c.

References [command_t](#), [PLACER](#), [st_cmp_ci_eq\(\)](#), and [st_cmp_ci_starts\(\)](#).

Referenced by [molten_parse\(\)](#), and [molten_sort\(\)](#).

5.294.1.2 void molten_parse (connection_t *con)

Definition at line 32 of file commands.c.

References [command](#), [command_t](#), [con_read_line\(\)](#), [enqueue\(\)](#), [molten_commands](#), [molten_compare\(\)](#), [molten_invalid\(\)](#), [molten_parse\(\)](#), [molten_quit\(\)](#), [pl_char_get\(\)](#), [pl_empty\(\)](#), and [pl_length_get\(\)](#).

Referenced by [molten_init\(\)](#), [molten_invalid\(\)](#), [molten_parse\(\)](#), and [molten_stats\(\)](#).

5.294.1.3 void molten_sort (void)

Sort the Molten command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 27 of file commands.c.

References [command_t](#), [molten_commands](#), and [molten_compare\(\)](#).

Referenced by [protocol_init\(\)](#).

5.295 src/servers/pop/commands.c File Reference

```
#include "magma.h"
#include "commands.h"
```

Functions

- [int_t pop_compare](#) (const void *compare, const void *command)
commands.c
- void [pop_sort](#) (void)
Sort the POP3 command table to be ready for binary searches.
- void [pop_requeue](#) (connection_t *con)
- void [pop_process](#) (connection_t *con)

5.295.1 Function Documentation

5.295.1.1 int_t pop_compare (const void * compare, const void * command)

commands.c

Definition at line 11 of file commands.c.

References [command_t](#), [PLACER](#), [st_cmp_ci_eq\(\)](#), and [st_cmp_ci_starts\(\)](#).

Referenced by [pop_process\(\)](#), and [pop_sort\(\)](#).

5.295.1.2 void pop_process (connection_t * con)

Definition at line 43 of file commands.c.

References [command](#), [command_t](#), [con_read_line\(\)](#), [enqueue\(\)](#), [pl_char_get\(\)](#), [pl_empty\(\)](#), [pl_length_get\(\)](#), [pop_commands](#), [pop_compare\(\)](#), [pop_invalid\(\)](#), [pop_process\(\)](#), [pop_quit\(\)](#), [pop_requeue\(\)](#), and [requeue\(\)](#).

Referenced by [pop_process\(\)](#), and [pop_requeue\(\)](#).

5.295.1.3 void pop_requeue (connection_t * con)

Definition at line 31 of file commands.c.

References [con_status\(\)](#), [enqueue\(\)](#), [pop_process\(\)](#), [pop_quit\(\)](#), and [status](#).

Referenced by [pop_init\(\)](#), and [pop_process\(\)](#).

5.295.1.4 void pop_sort (void)

Sort the POP3 command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 26 of file commands.c.

References [command_t](#), [pop_commands](#), and [pop_compare\(\)](#).

Referenced by [protocol_init\(\)](#).

5.296 src/servers/smtp/commands.c File Reference

```
#include "magma.h"
#include "commands.h"
```

Functions

- [int_t smtp_compare](#) (const void *compare, const void *[command](#))
- void [smtp_sort](#) (void)
Sort the SMTP command table to be ready for binary searches.
- void [smtp_requeue](#) ([connection_t](#) *con)
commands.c
- void [smtp_process](#) ([connection_t](#) *con)
The main entry point in the SMTP server for processing commands issued by clients.

5.296.1 Function Documentation

5.296.1.1 int_t smtp_compare (const void * *compare*, const void * *command*)

Definition at line 11 of file `commands.c`.

References `command_t`, `PLACER`, `st_cmp_ci_eq()`, and `st_cmp_ci_starts()`.

Referenced by `smtp_process()`, and `smtp_sort()`.

5.296.1.2 void smtp_process (connection_t * *con*)

The main entry point in the SMTP server for processing commands issued by clients.

Parameters:

con a pointer to the connection object of the client issuing the SMTP command.

Returns:

This function returns no value.

Definition at line 48 of file `commands.c`.

References `command`, `command_t`, `con_read_line()`, `enqueue()`, `pl_char_get()`, `pl_empty()`, `pl_length_get()`, `requeue()`, `smtp_commands`, `smtp_compare()`, `smtp_data()`, `smtp_invalid()`, `smtp_process()`, `smtp_quit()`, and `smtp_requeue()`.

Referenced by `smtp_process()`, and `smtp_requeue()`.

5.296.1.3 void smtp_requeue (connection_t * *con*)

`commands.c`

Definition at line 31 of file `commands.c`.

References `con_status()`, `enqueue()`, `smtp_process()`, `smtp_quit()`, and `status`.

Referenced by `smtp_data()`, `smtp_init()`, and `smtp_process()`.

5.296.1.4 void smtp_sort (void)

Sort the SMTP command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 26 of file commands.c.

References `command_t`, `smtp_commands`, and `smtp_compare()`.

Referenced by `protocol_init()`.

5.297 src/providers/dime/dmtp/commands.h File Reference

```
#include <stdlib.h>
#include "dime/sds/sds.h"
```

Data Structures

- struct [dmtp_argument_t](#)
- struct [dmtp_command_t](#)
- struct [dmtp_command_key_t](#)

Defines

- #define [DMTP_MAX_ARGUMENT_NUM](#) 3
- #define [DMTP_COMMANDS_NUM](#) 14

Enumerations

- enum [dmtp_command_type_t](#) {
[DMTP_STARTTLS](#) = 0, [DMTP_HELO](#), [DMTP_EHLO](#), [DMTP_MODE](#),
[DMTP_RSET](#), [DMTP_NOOP](#), [DMTP_HELP](#), [DMTP_QUIT](#),
[DMTP_MAIL](#), [DMTP_RCPT](#), [DMTP_DATA](#), [DMTP_SGNT](#),
[DMTP_HIST](#), [DMTP_VRFY](#), [DMTP_COMMAND_INVALID](#) }
- enum [dmtp_argument_type_t](#) { [DMTP_ARG_NONE](#) = 0, [DMTP_ARG_PLAIN](#), [DMTP_ARG_ANGULAR_BRCKT](#), [DMTP_ARG_SQUARE_BRCKT](#) }
- enum [dmtp_mode_t](#) {
[DMTP_MODE_DMTP](#), [DMTP_MODE_SMTP](#), [DMTP_MODE_NONE](#), [dmtp_mode_unknown](#) = 0,
[dmtp_mode_dual](#) = 1, [dmtp_mode_dmtp](#) = 2, [dmtp_mode_smtp](#) = 3, [dmtp_mode_esmtp](#) = 4 }
- enum [dmtp_parse_state_t](#) {
[DMTP_PARSE_SUCCESS](#), [DMTP_PARSE_INVALID_CALL](#), [DMTP_PARSE_INTERNAL_ERROR](#), [DMTP_PARSE_COMMAND_ERROR](#),
[DMTP_PARSE_ARGUMENT_ERROR](#) }

Functions

- [dmtp_command_t * dime_dmtp_command_create](#) ([dmtp_command_type_t](#) type)
Create a dmtp command structure.
- void [dime_dmtp_command_destroy](#) ([dmtp_command_t *command](#))
Destroy a dmtp command structure.
- [sds dime_dmtp_command_format](#) ([dmtp_command_t *command](#))
Formats the specified dmtp command into a sds string.
- [dmtp_parse_state_t dime_dmtp_command_parse](#) ([sds comm_line](#), [dmtp_command_t **comm_struct](#))
Parse the provided sds string into a dmtp command structure.
- [sds dime_dmtp_command_starttls](#) ([sds host](#), [dmtp_mode_t mode](#))
Create a string containing dmtp STARTTLS command with specified arguments.

- [sds dime_dmtp_command_helo](#) ([sds](#) host)
Create a string containing dmtp HELO command with specified arguments.
- [sds dime_dmtp_command_ehlo](#) ([sds](#) host)
Create a string containing dmtp EHLO command with specified arguments.
- [sds dime_dmtp_command_mode](#) ([void](#))
Create a string containing dmtp MODE command with specified arguments.
- [sds dime_dmtp_command_rset](#) ([void](#))
Create a string containing dmtp RSET command with specified arguments.
- [sds dime_dmtp_command_noop](#) ([sds](#) optarg1, [sds](#) optarg2, [sds](#) optarg3)
Create a string containing dmtp NOOP command with specified arguments.
- [sds dime_dmtp_command_help](#) ([void](#))
Create a string containing dmtp HELP command with specified arguments.
- [sds dime_dmtp_command_quit](#) ([void](#))
Create a string containing dmtp QUIT command with specified arguments.
- [sds dime_dmtp_command_mail](#) ([sds](#) from, [sds](#) fingerprint)
Create a string containing dmtp MAIL command with specified arguments.
- [sds dime_dmtp_command_rcpt](#) ([sds](#) to, [sds](#) fingerprint)
Create a string containing dmtp RCPT command with specified arguments.
- [sds dime_dmtp_command_data](#) ([void](#))
Create a string containing dmtp DATA command with specified arguments.
- [sds dime_dmtp_command_sgmt_user](#) ([sds](#) address, [sds](#) fingerprint)
Create a string containing dmtp SGNT command with specified arguments.
- [sds dime_dmtp_command_sgmt_domain](#) ([sds](#) domain, [sds](#) fingerprint)
Create a string containing dmtp SGNT command with specified arguments.
- [sds dime_dmtp_command_hist](#) ([sds](#) address, [sds](#) start, [sds](#) stop)
Create a string containing dmtp HIST command with specified arguments.
- [sds dime_dmtp_command_vrfy_user](#) ([sds](#) address, [sds](#) fingerprint)
Create a string containing dmtp VRFY command with specified arguments.
- [sds dime_dmtp_command_vrfy_domain](#) ([sds](#) domain, [sds](#) fingerprint)
Create a string containing dmtp VRFY command with specified arguments.

Variables

- [dmtp_command_key_t dmtp_command_list](#) [DMTP_COMMANDS_NUM]

5.297.1 Define Documentation

5.297.1.1 #define DMTP_COMMANDS_NUM 14

Definition at line 8 of file commands.h.

5.297.1.2 #define DMTP_MAX_ARGUMENT_NUM 3

Definition at line 7 of file commands.h.

5.297.2 Enumeration Type Documentation

5.297.2.1 enum dmtplib_argument_type_t

Enumerator:

DMTP_ARG_NONE

DMTP_ARG_PLAIN

DMTP_ARG_ANGULAR_BRCKT

DMTP_ARG_SQUARE_BRCKT

Definition at line 28 of file commands.h.

5.297.2.2 enum dmtplib_command_type_t

Enumerator:

DMTP_STARTTLS

DMTP_HELO

DMTP_EHLO

DMTP_MODE

DMTP_RSET

DMTP_NOOP

DMTP_HELP

DMTP_QUIT

DMTP_MAIL

DMTP_RCPT

DMTP_DATA

DMTP_SGNT

DMTP_HIST

DMTP_VRFY

DMTP_COMMAND_INVALID

Definition at line 10 of file commands.h.

5.297.2.3 enum dmtp_mode_t

Enumerator:

DMTP_MODE_DMTP
DMTP_MODE_SMTP
DMTP_MODE_NONE
dmtp_mode_unknown
dmtp_mode_dual
dmtp_mode_dmtp
dmtp_mode_smtp
dmtp_mode_esmtp

Definition at line 35 of file commands.h.

5.297.2.4 enum dmtp_parse_state_t

Enumerator:

DMTP_PARSE_SUCCESS
DMTP_PARSE_INVALID_CALL
DMTP_PARSE_INTERNAL_ERROR
DMTP_PARSE_COMMAND_ERROR
DMTP_PARSE_ARGUMENT_ERROR

Definition at line 41 of file commands.h.

5.297.3 Function Documentation

5.297.3.1 dmtp_command_t* dime_dmtp_command_create(dmtp_command_type_t type)

Create a dmtp command structure.

Parameters:

type dmtp command enum.

Returns:

A newly created dmtp command structure with specified type and NULL args. dime_dmtp_command_destroy

Definition at line 1086 of file commands.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.297.3.2 sds dime_dmtp_command_data(void)

Create a string containing dmtp DATA command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1671 of file commands.c.

References DMTP_DATA, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.297.3.3 void dime_dmtplib_command_destroy (dmtplib_command_t * *command*)

Destroy a dmtplib command structure.

Parameters:

command dmtplib command structure to be destroyed.

Definition at line 1099 of file commands.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.297.3.4 sds dime_dmtplib_command_ehlo (sds *host*)

Create a string containing dmtplib EHLO command with specified arguments.

Parameters:

host Name of the domain that is being connected to.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1283 of file commands.c.

References dmtplib_command_t::args, DMTP_EHLO, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.5 sds dime_dmtplib_command_format (dmtplib_command_t * *command*)

Formats the specified dmtplib command into a sds string.

Parameters:

command Command to be formatted.

Returns:

sds string containing formatted command. sdsfree

Definition at line 1116 of file commands.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.297.3.6 sds dime_dmtplib_command_helo (sds *host*)

Create a string containing dmtplib HELO command with specified arguments.

Parameters:

host Name of the domain that is being connected to.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1232 of file commands.c.

References dmtplib_command_t::args, DMTP_HELO, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.7 sds dime_dmtp_command_help (void)

Create a string containing dmtp HELP command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1468 of file commands.c.

References DMTP_HELP, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.297.3.8 sds dime_dmtp_command_hist (sds address, sds start, sds stop)

Create a string containing dmtp HIST command with specified arguments.

Parameters:

address sds string containing user mail address.

start sds string containing optional starting user signet fingerprint. NULL if unwanted.

stop sds string containing optional ending user signet fingerprint. NULL if unwanted.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1832 of file commands.c.

References dmtp_command_t::args, DMTP_HIST, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.9 sds dime_dmtp_command_mail (sds from, sds fingerprint)

Create a string containing dmtp MAIL command with specified arguments.

Parameters:

from sds string containg origin domain.

fingerprint sds string containing origin signet full fingerprint.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1547 of file commands.c.

References dmtp_command_t::args, DMTP_MAIL, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.10 sds dime_dmtp_command_mode (void)

Create a string containing dmtp MODE command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1332 of file commands.c.

References DMTP_MODE, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.297.3.11 `sds dime_dmtplib_command_noop (sds optarg1, sds optarg2, sds optarg3)`

Create a string containing dmtplib NOOP command with specified arguments.

Parameters:

optarg1 Optional allowed argument 1. NULL if unwanted.

optarg2 Optional allowed argument 2. NULL if unwanted.

optarg3 Optional allowed argument 3. NULL if unwanted.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1412 of file commands.c.

References dmtplib_command_t::args, DMTP_NOOP, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.12 `dmtplib_parse_state_t dime_dmtplib_command_parse (sds comm_line, dmtplib_command_t ** comm_struct)`

Parse the provided sds string into a dmtplib command structure.

Parameters:

command sds dmtplib command string.

Returns:

A valid dmtplib command structure on success, NULL on failure. dime_dmtplib_command_destroy

Definition at line 1133 of file commands.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.297.3.13 `sds dime_dmtplib_command_quit (void)`

Create a string containing dmtplib QUIT command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1505 of file commands.c.

References DMTP_QUIT, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.297.3.14 `sds dime_dmtplib_command_rcpt (sds to, sds fingerprint)`

Create a string containing dmtplib RCPT command with specified arguments.

Parameters:

to sds string containing destination domain.

fingerprint sds string containing destination signet full fingerprint.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1611 of file commands.c.

References dmtplib_command_t::args, DMTP_RCPT, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.15 sds dime_dmtp_command_rset (void)

Create a string containing dmtp RSET command with specified arguments.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1369 of file commands.c.

References DMTP_RSET, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, and PUSH_ERROR.

5.297.3.16 sds dime_dmtp_command_sgnt_domain (sds domain, sds fingerprint)

Create a string containing dmtp SGNT command with specified arguments.

Parameters:

domain sds string containing domain name.

fingerprint sds string containing optional organizational signet fingerprint. NULL if unwanted.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1771 of file commands.c.

References dmtp_command_t::args, DMTP_SGNT, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.17 sds dime_dmtp_command_sgnt_user (sds address, sds fingerprint)

Create a string containing dmtp SGNT command with specified arguments.

Parameters:

address sds string containing mail address of the user.

fingerprint sds string containing optional user signet fingerprint. NULL if unwanted.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1712 of file commands.c.

References dmtp_command_t::args, DMTP_SGNT, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.18 sds dime_dmtp_command_starttls (sds host, dmtp_mode_t mode)

Create a string containing dmtp STARTTLS command with specified arguments.

Parameters:

host Name of the domain that is being connected to.

mode optional mode parameter (DMTP or SMTP). DMTP_MODE_NONE to not specify mode.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1154 of file commands.c.

References dmtp_command_t::args, DMTP_MODE_DMTP, DMTP_MODE_NONE, DMTP_MODE_SMTP, DMTP_STARTTLS, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, sdsdup(), and sdsnewlen().

5.297.3.19 sds dime_dmtp_command_vrfy_domain (sds domain, sds fingerprint)

Create a string containing dmtp VRFY command with specified arguments.

Parameters:

domain sds string containing domain name.

start sds string containing organizational signet fingerprint.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1961 of file commands.c.

References dmtp_command_t::args, DMTP_VRFY, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.3.20 sds dime_dmtp_command_vrfy_user (sds address, sds fingerprint)

Create a string containing dmtp VRFY command with specified arguments.

Parameters:

address sds string containing user mail address.

start sds string containing user signet fingerprint.

Returns:

sds string containing the desired command. sdsfree

Definition at line 1897 of file commands.c.

References dmtp_command_t::args, DMTP_VRFY, ERR_BAD_PARAM, ERR_UNSPEC, PUBLIC_FUNC_PROLOGUE, PUSH_ERROR, and sdsdup().

5.297.4 Variable Documentation**5.297.4.1 dmtp_command_key_t dmtp_command_list[DMTP_COMMANDS_NUM]**

Definition at line 2013 of file commands.c.

5.298 src/servers/dmtp/commands.h File Reference

Variables

- [command_t dmtp_commands](#) []

5.298.1 Variable Documentation

5.298.1.1 [command_t dmtp_commands](#) []

Initial value:

```
{

    { .string = "EHLO", .length = 4, .function = &dmtp_ehlo },
    { .string = "HELO", .length = 4, .function = &dmtp_helo },
    { .string = "NOOP", .length = 4, .function = &dmtp_noop },
    { .string = "MODE", .length = 4, .function = &dmtp_mode },
    { .string = "RSET", .length = 4, .function = &dmtp_rset },
    { .string = "QUIT", .length = 4, .function = &dmtp_quit },

    { .string = "MAIL", .length = 9, .function = &dmtp_mail },
    { .string = "RCPT", .length = 7, .function = &dmtp_rcpt },
    { .string = "DATA", .length = 4, .function = &dmtp_data },

    { .string = "SGNT", .length = 9, .function = &dmtp_sgnt },
    { .string = "HIST", .length = 7, .function = &dmtp_hist },
    { .string = "VRFY", .length = 4, .function = &dmtp_vrfy },

    { .string = "HELP", .length = 4, .function = &dmtp_help },
    { .string = "VERB", .length = 4, .function = &dmtp_verb }
}
```

Definition at line 11 of file `commands.h`.

Referenced by `dmtp_process()`, and `dmtp_sort()`.

5.299 src/servers/http/commands.h File Reference

5.300 src/servers/imap/commands.h File Reference

Variables

- [command_t imap_commands](#) []

5.300.1 Variable Documentation

5.300.1.1 command_t imap_commands[]

Initial value:

```
{
{ .string = "ID", .length = 2, .function = &imap_id},
{ .string = "COPY", .length = 4, .function = &imap_copy},
{ .string = "IDLE", .length = 4, .function = &imap_idle},
{ .string = "LIST", .length = 4, .function = &imap_list},
{ .string = "LSUB", .length = 4, .function = &imap_lsub},
{ .string = "NOOP", .length = 4, .function = &imap_noop},
{ .string = "CHECK", .length = 5, .function = &imap_check},
{ .string = "CLOSE", .length = 5, .function = &imap_close},
{ .string = "FETCH", .length = 5, .function = &imap_fetch},
{ .string = "LOGIN", .length = 5, .function = &imap_login},
{ .string = "STORE", .length = 5, .function = &imap_store},
{ .string = "APPEND", .length = 6, .function = &imap_append},
{ .string = "CREATE", .length = 6, .function = &imap_create},
{ .string = "DELETE", .length = 6, .function = &imap_delete},
{ .string = "RENAME", .length = 6, .function = &imap_rename},
{ .string = "SEARCH", .length = 6, .function = &imap_search},
{ .string = "SELECT", .length = 6, .function = &imap_select},
{ .string = "LOGOUT", .length = 6, .function = &imap_logout},
{ .string = "STATUS", .length = 6, .function = &imap_status},
{ .string = "EXAMINE", .length = 7, .function = &imap_examine},
{ .string = "EXPUNGE", .length = 7, .function = &imap_expunge},
{ .string = "STARTTLS", .length = 8, .function = &imap_starttls},
{ .string = "SUBSCRIBE", .length = 9, .function = &imap_subscribe},
{ .string = "CAPABILITY", .length = 10, .function = &imap_capability},
{ .string = "UNSUBSCRIBE", .length = 11, .function = &imap_unsubscribe}
}
```

Definition at line 11 of file commands.h.

Referenced by `imap_process()`, and `imap_sort()`.

5.301 src/servers/molten/commands.h File Reference

Variables

- [command_t molten_commands](#) []

5.301.1 Variable Documentation

5.301.1.1 [command_t molten_commands](#)[]

Initial value:

```
{
    {
        .string = "QUIT",
        .length = 4,
        .function = &molten_quit
    }, {
        .string = "STATS",
        .length = 5,
        .function = &molten_stats
    }
}
```

Definition at line 11 of file `commands.h`.

Referenced by `molten_parse()`, and `molten_sort()`.

5.302 src/servers/pop/commands.h File Reference

Variables

- [command_t pop_commands](#) []

5.302.1 Variable Documentation

5.302.1.1 `command_t pop_commands` []

Definition at line 12 of file `commands.h`.

Referenced by `pop_process()`, and `pop_sort()`.

5.303 src/servers/smtp/commands.h File Reference

Variables

- [command_t smtp_commands](#) []

5.303.1 Variable Documentation

5.303.1.1 `command_t smtp_commands` []

Definition at line 11 of file `commands.h`.

Referenced by `smtp_process()`, and `smtp_sort()`.

5.304 src/providers/dime/ed25519/curve25519-donna-32bit.h File Reference

Defines

- #define `curve25519_mul_noinline` `curve25519_mul`
- #define `F(s)`
- #define `carry_pass()`
- #define `carry_pass_full()`
- #define `carry_pass_final()`
- #define `F(i, s)`

Typedefs

- typedef uint32_t `bignum25519` [10]
- typedef uint32_t `bignum25519align16` [12]

5.304.1 Define Documentation

5.304.1.1 #define `carry_pass()`

Value:

```
f[1] += f[0] >> 26; f[0] &= reduce_mask_26; \
f[2] += f[1] >> 25; f[1] &= reduce_mask_25; \
f[3] += f[2] >> 26; f[2] &= reduce_mask_26; \
f[4] += f[3] >> 25; f[3] &= reduce_mask_25; \
f[5] += f[4] >> 26; f[4] &= reduce_mask_26; \
f[6] += f[5] >> 25; f[5] &= reduce_mask_25; \
f[7] += f[6] >> 26; f[6] &= reduce_mask_26; \
f[8] += f[7] >> 25; f[7] &= reduce_mask_25; \
f[9] += f[8] >> 26; f[8] &= reduce_mask_26;
```

5.304.1.2 #define `carry_pass_final()`

Value:

```
carry_pass() \
    f[9] &= reduce_mask_25;
```

5.304.1.3 #define `carry_pass_full()`

Value:

```
carry_pass() \
    f[0] += 19 * (f[9] >> 25); f[9] &= reduce_mask_25;
```

5.304.1.4 #define `curve25519_mul_noinline` `curve25519_mul`

Definition at line 152 of file `curve25519-donna-32bit.h`.

5.304.1.5 #define F(i, s)

Value:

```
out[s+0] |= (unsigned char )(f[i] & 0xff); \  
out[s+1] = (unsigned char )((f[i] >> 8) & 0xff); \  
out[s+2] = (unsigned char )((f[i] >> 16) & 0xff); \  
out[s+3] = (unsigned char )((f[i] >> 24) & 0xff);
```

5.304.1.6 #define F(s)

Value:

```
((((uint32_t)in[s + 0])          ) | \  
  (((uint32_t)in[s + 1]) << 8) | \  
  (((uint32_t)in[s + 2]) << 16) | \  
  (((uint32_t)in[s + 3]) << 24))
```

5.304.2 Typedef Documentation

5.304.2.1 typedef uint32_t bignum25519[10]

Definition at line 8 of file curve25519-donna-32bit.h.

5.304.2.2 typedef uint32_t bignum25519align16[12]

Definition at line 9 of file curve25519-donna-32bit.h.

5.305 src/providers/dime/ed25519/curve25519-donna-64bit.h File Reference

Defines

- #define [F\(s\)](#)
- #define [curve25519_contract_carry\(\)](#)
- #define [curve25519_contract_carry_full\(\)](#)
- #define [curve25519_contract_carry_final\(\)](#)
- #define [write51full\(n, shift\)](#)
- #define [write51\(n\)](#) [write51full\(n,13*n\)](#)
- #define [ED25519_64BIT_TABLES](#)

Typedefs

- typedef uint64_t [bignum25519](#) [5]

5.305.1 Define Documentation

5.305.1.1 #define curve25519_contract_carry()

Value:

```
t[1] += t[0] >> 51; t[0] &= reduce_mask_51; \
t[2] += t[1] >> 51; t[1] &= reduce_mask_51; \
t[3] += t[2] >> 51; t[2] &= reduce_mask_51; \
t[4] += t[3] >> 51; t[3] &= reduce_mask_51;
```

5.305.1.2 #define curve25519_contract_carry_final()

Value:

```
curve25519_contract_carry() \
t[4] &= reduce_mask_51;
```

5.305.1.3 #define curve25519_contract_carry_full()

Value:

```
curve25519_contract_carry() \
t[0] += 19 * (t[4] >> 51); t[4] &= reduce_mask_51;
```

5.305.1.4 #define ED25519_64BIT_TABLES

Definition at line 412 of file curve25519-donna-64bit.h.

5.305.1.5 #define F(s)

Value:

```

(((uint64_t)in[s + 0])      ) | \
  (((uint64_t)in[s + 1]) << 8) | \
  (((uint64_t)in[s + 2]) << 16) | \
  (((uint64_t)in[s + 3]) << 24) | \
  (((uint64_t)in[s + 4]) << 32) | \
  (((uint64_t)in[s + 5]) << 40) | \
  (((uint64_t)in[s + 6]) << 48) | \
  (((uint64_t)in[s + 7]) << 56))

```

5.305.1.6 #define write51(n) write51full(n,13*n)

5.305.1.7 #define write51full(n, shift)

Value:

```

f = ((t[n] >> shift) | (t[n+1] << (51 - shift))); \
  for (i = 0; i < 8; i++, f >>= 8) *out++ = (unsigned char)f;

```

5.305.2 Typedef Documentation

5.305.2.1 typedef uint64_t bignum25519[5]

Definition at line 9 of file curve25519-donna-64bit.h.

5.306 src/providers/dime/ed25519/curve25519-donna-helpers.h File Reference

5.307 src/providers/dime/ed25519/curve25519-donna-sse2.h File Reference

```
#include <emmintrin.h>
```

Data Structures

- union [packedelem8_t](#)
- union [packedelem32_t](#)
- union [packedelem64_t](#)

Defines

- #define [curve25519_add_after_basic](#) curve25519_add_reduce
- #define [curve25519_square](#)(r, n) curve25519_square_times(r, n, 1)
- #define [carry_pass](#)()
- #define [carry_pass_full](#)()
- #define [carry_pass_final](#)()
- #define [F](#)(i, s)

Typedefs

- typedef __m128i [xmimi](#)
- typedef union [packedelem8_t](#) [packedelem8](#)
- typedef union [packedelem32_t](#) [packedelem32](#)
- typedef union [packedelem64_t](#) [packedelem64](#)
- typedef uint32_t [bignum25519](#) [12]
- typedef [packedelem32](#) [packed32bignum25519](#) [5]
- typedef [packedelem64](#) [packed64bignum25519](#) [10]

5.307.1 Define Documentation

5.307.1.1 #define carry_pass()

Value:

```
f[1] += f[0] >> 26; f[0] &= 0x3fffffff; \
    f[2] += f[1] >> 25; f[1] &= 0x1fffffff; \
    f[3] += f[2] >> 26; f[2] &= 0x3fffffff; \
    f[4] += f[3] >> 25; f[3] &= 0x1fffffff; \
    f[5] += f[4] >> 26; f[4] &= 0x3fffffff; \
    f[6] += f[5] >> 25; f[5] &= 0x1fffffff; \
    f[7] += f[6] >> 26; f[6] &= 0x3fffffff; \
    f[8] += f[7] >> 25; f[7] &= 0x1fffffff; \
    f[9] += f[8] >> 26; f[8] &= 0x3fffffff;
```

5.307.1.2 #define carry_pass_final()

Value:

```
carry_pass() \
    f[9] &= 0x1fffffff;
```

5.307.1.3 #define carry_pass_full()**Value:**

```
carry_pass() \
    f[0] += 19 * (f[9] >> 25); f[9] &= 0xffffffff;
```

5.307.1.4 #define curve25519_add_after_basic curve25519_add_reduce

Definition at line 99 of file curve25519-donna-sse2.h.

5.307.1.5 #define curve25519_square(r, n) curve25519_square_times(r, n, 1)

Definition at line 431 of file curve25519-donna-sse2.h.

5.307.1.6 #define F(i, s)**Value:**

```
out[s+0] |= (unsigned char )(f[i] & 0xff); \
    out[s+1] = (unsigned char )((f[i] >> 8) & 0xff); \
    out[s+2] = (unsigned char )((f[i] >> 16) & 0xff); \
    out[s+3] = (unsigned char )((f[i] >> 24) & 0xff);
```

5.307.2 Typedef Documentation**5.307.2.1 typedef uint32_t bignum25519[12]**

Definition at line 27 of file curve25519-donna-sse2.h.

5.307.2.2 typedef packedelem32 packed32bignum25519[5]

Definition at line 28 of file curve25519-donna-sse2.h.

5.307.2.3 typedef packedelem64 packed64bignum25519[10]

Definition at line 29 of file curve25519-donna-sse2.h.

5.307.2.4 typedef union packedelem32_t packedelem32**5.307.2.5 typedef union packedelem64_t packedelem64****5.307.2.6 typedef union packedelem8_t packedelem8****5.307.2.7 typedef __m128i xmmi**

Definition at line 9 of file curve25519-donna-sse2.h.

5.308 src/providers/dime/ed25519/ed25519-donna-32bit-sse2.h File Reference

5.309 src/providers/dime/ed25519/ed25519-donna-32bit-tables.h File Reference

5.310 `src/providers/dime/ed25519/ed25519-donna-64bit-sse2.h` File Reference

5.311 src/providers/dime/ed25519/ed25519-donna-64bit-tables.h File Reference

5.312 `src/providers/dime/ed25519/ed25519-donna-64bit-x86-32bit.h` File Reference

5.313 src/providers/dime/ed25519/ed25519-donna-64bit-x86.h File Reference

5.314 `src/providers/dime/ed25519/ed25519-donna-basepoint-table.h` File Reference

5.315 src/providers/dime/ed25519/ed25519-donna-batchverify.h File Reference

Data Structures

- struct [batch_heap_t](#)

Defines

- #define [max_batch_size](#) 64
- #define [heap_batch_size](#) ((max_batch_size * 2) + 1)

Typedefs

- typedef size_t [heap_index_t](#)
- typedef struct [batch_heap_t](#) [batch_heap](#)

Functions

- int ED25519_FN() [ed25519_sign_open_batch](#) (const unsigned char **m, size_t *mlen, const unsigned char **pk, const unsigned char **RS, size_t num, int *valid)

Variables

- unsigned char [batch_point_buffer](#) [3][32]

5.315.1 Define Documentation

5.315.1.1 #define heap_batch_size ((max_batch_size * 2) + 1)

Definition at line 6 of file ed25519-donna-batchverify.h.

5.315.1.2 #define max_batch_size 64

Definition at line 5 of file ed25519-donna-batchverify.h.

Referenced by [ed25519_sign_open_batch\(\)](#).

5.315.2 Typedef Documentation

5.315.2.1 typedef struct batch_heap_t batch_heap

5.315.2.2 typedef size_t heap_index_t

Definition at line 11 of file ed25519-donna-batchverify.h.

5.315.3 Function Documentation

5.315.3.1 int ED25519_FN() ed25519_sign_open_batch (const unsigned char ** m, size_t * mlen, const unsigned char ** pk, const unsigned char ** RS, size_t num, int * valid)

Definition at line 205 of file ed25519-donna-batchverify.h.

References ALIGN, ED25519_FN, ed25519_randombytes_unsafe(), ed25519_sign_open(), max_batch_size, batch_heap_t::points, batch_heap_t::r, and batch_heap_t::scalars.

5.315.4 Variable Documentation

5.315.4.1 unsigned char batch_point_buffer[3][32]

Definition at line 191 of file ed25519-donna-batchverify.h.

5.316 src/providers/dime/ed25519/ed25519-donna-impl-base.h File Reference

Defines

- #define [S1_SWINDOWSIZE](#) 5
- #define [S1_TABLE_SIZE](#) (1<<(S1_SWINDOWSIZE-2))
- #define [S2_SWINDOWSIZE](#) 7
- #define [S2_TABLE_SIZE](#) (1<<(S2_SWINDOWSIZE-2))

5.316.1 Define Documentation

5.316.1.1 #define S1_SWINDOWSIZE 5

Definition at line 246 of file ed25519-donna-impl-base.h.

5.316.1.2 #define S1_TABLE_SIZE (1<<(S1_SWINDOWSIZE-2))

Definition at line 247 of file ed25519-donna-impl-base.h.

5.316.1.3 #define S2_SWINDOWSIZE 7

Definition at line 248 of file ed25519-donna-impl-base.h.

5.316.1.4 #define S2_TABLE_SIZE (1<<(S2_SWINDOWSIZE-2))

Definition at line 249 of file ed25519-donna-impl-base.h.

5.317 src/providers/dime/ed25519/ed25519-donna-impl-sse2.h File Reference

Defines

- #define [S1_SWINDOWSIZE](#) 5
- #define [S1_TABLE_SIZE](#) (1<<(S1_SWINDOWSIZE-2))
- #define [S2_SWINDOWSIZE](#) 7
- #define [S2_TABLE_SIZE](#) (1<<(S2_SWINDOWSIZE-2))

5.317.1 Define Documentation

5.317.1.1 #define S1_SWINDOWSIZE 5

Definition at line 278 of file ed25519-donna-impl-sse2.h.

5.317.1.2 #define S1_TABLE_SIZE (1<<(S1_SWINDOWSIZE-2))

Definition at line 279 of file ed25519-donna-impl-sse2.h.

5.317.1.3 #define S2_SWINDOWSIZE 7

Definition at line 280 of file ed25519-donna-impl-sse2.h.

5.317.1.4 #define S2_TABLE_SIZE (1<<(S2_SWINDOWSIZE-2))

Definition at line 281 of file ed25519-donna-impl-sse2.h.

5.318 src/providers/dime/ed25519/ed25519-donna-portable-identify.h File Reference

```
#include <sys/param.h>
#include <stdint.h>
```

Defines

- `#define` [OS_NIX](#)

5.318.1 Define Documentation

5.318.1.1 `#define` OS_NIX

Definition at line 8 of file ed25519-donna-portable-identify.h.

5.319 src/providers/dime/ed25519/ed25519-donna-portable.h File Reference

```
#include "../ed25519/ed25519-donna-portable-identify.h"
#include <sys/param.h>
#include <stdlib.h>
#include <string.h>
```

Defines

- #define [mul32x32_64](#)(a, b) (((uint64_t)(a))*(b))
- #define [DONNA_INLINE](#) inline [__attribute__\(\(always_inline\)\)](#)
- #define [DONNA_NOINLINE](#) [__attribute__\(\(noinline\)\)](#)
- #define [ALIGN](#)(x) [__attribute__\(\(aligned\(x\)\)\)](#)
- #define [ROTL32](#)(a, b) (((a) << (b)) | ((a) >> (32 - b)))
- #define [ROTR32](#)(a, b) (((a) >> (b)) | ((a) << (32 - b)))

5.319.1 Define Documentation

5.319.1.1 #define [ALIGN](#)(x) [__attribute__\(\(aligned\(x\)\)\)](#)

Definition at line 24 of file ed25519-donna-portable.h.

Referenced by [curved25519_scalarmult_basepoint\(\)](#), [ed25519_publickey\(\)](#), [ed25519_sign\(\)](#), [ed25519_sign_open\(\)](#), and [ed25519_sign_open_batch\(\)](#).

5.319.1.2 #define [DONNA_INLINE](#) inline [__attribute__\(\(always_inline\)\)](#)

Definition at line 21 of file ed25519-donna-portable.h.

5.319.1.3 #define [DONNA_NOINLINE](#) [__attribute__\(\(noinline\)\)](#)

Definition at line 22 of file ed25519-donna-portable.h.

5.319.1.4 #define [mul32x32_64](#)(a, b) (((uint64_t)(a))*(b))

Definition at line 3 of file ed25519-donna-portable.h.

5.319.1.5 #define [ROTL32](#)(a, b) (((a) << (b)) | ((a) >> (32 - b)))

Definition at line 25 of file ed25519-donna-portable.h.

5.319.1.6 #define [ROTR32](#)(a, b) (((a) >> (b)) | ((a) << (32 - b)))

Definition at line 26 of file ed25519-donna-portable.h.

5.320 src/providers/dime/ed25519/ed25519-donna.h File Reference

```
#include "../ed25519/ed25519-donna-portable.h"
#include "../ed25519/curve25519-donna-32bit.h"
#include "../ed25519/curve25519-donna-helpers.h"
#include "../ed25519/modm-donna-32bit.h"
#include "../ed25519/ed25519-donna-basepoint-table.h"
#include "../ed25519/ed25519-donna-32bit-tables.h"
#include "../ed25519/ed25519-donna-64bit-x86-32bit.h"
#include "../ed25519/ed25519-donna-impl-base.h"
```

Data Structures

- struct [ge25519_t](#)
- struct [ge25519_p1p1_t](#)
- struct [ge25519_niels_t](#)
- struct [ge25519_pniels_t](#)

Defines

- #define [ED25519_32BIT](#)

Typedefs

- typedef unsigned char [hash_512bits](#) [64]
- typedef struct [ge25519_t](#) [ge25519](#)
- typedef struct [ge25519_p1p1_t](#) [ge25519_p1p1](#)
- typedef struct [ge25519_niels_t](#) [ge25519_niels](#)
- typedef struct [ge25519_pniels_t](#) [ge25519_pniels](#)

5.320.1 Define Documentation

5.320.1.1 #define ED25519_32BIT

Definition at line 19 of file ed25519-donna.h.

5.320.2 Typedef Documentation

5.320.2.1 typedef struct ge25519_t ge25519

5.320.2.2 typedef struct ge25519_niels_t ge25519_niels

5.320.2.3 typedef struct ge25519_p1p1_t ge25519_p1p1

5.320.2.4 typedef struct ge25519_pniels_t ge25519_pniels

5.320.2.5 typedef unsigned char hash_512bits[64]

Definition at line 61 of file ed25519-donna.h.

5.321 src/providers/dime/ed25519/fuzz/ed25519-donna.h File Reference

```
#include <stdlib.h>
```

Typedefs

- typedef unsigned char [ed25519_signature](#) [64]
- typedef unsigned char [ed25519_public_key](#) [32]
- typedef unsigned char [ed25519_secret_key](#) [32]
- typedef unsigned char [curved25519_key](#) [32]

Functions

- void [ed25519_publickey](#) (const [ed25519_secret_key](#) sk, [ed25519_public_key](#) pk)
- int [ed25519_sign_open](#) (const unsigned char *m, size_t mlen, const [ed25519_public_key](#) pk, const [ed25519_signature](#) RS)
- void [ed25519_sign](#) (const unsigned char *m, size_t mlen, const [ed25519_secret_key](#) sk, const [ed25519_public_key](#) pk, [ed25519_signature](#) RS)
- int [ed25519_sign_open_batch](#) (const unsigned char **m, size_t *mlen, const unsigned char **pk, const unsigned char **RS, size_t num, int *valid)
- void [ed25519_randombytes_unsafe](#) (void *out, size_t count)
- void [curved25519_scalarmult_basepoint](#) ([curved25519_key](#) pk, const [curved25519_key](#) e)
- void [ed25519_publickey_sse2](#) (const [ed25519_secret_key](#) sk, [ed25519_public_key](#) pk)
- int [ed25519_sign_open_sse2](#) (const unsigned char *m, size_t mlen, const [ed25519_public_key](#) pk, const [ed25519_signature](#) RS)
- void [ed25519_sign_sse2](#) (const unsigned char *m, size_t mlen, const [ed25519_secret_key](#) sk, const [ed25519_public_key](#) pk, [ed25519_signature](#) RS)
- int [ed25519_sign_open_batch_sse2](#) (const unsigned char **m, size_t *mlen, const unsigned char **pk, const unsigned char **RS, size_t num, int *valid)
- void [ed25519_randombytes_unsafe_sse2](#) (void *out, size_t count)
- void [curved25519_scalarmult_basepoint_sse2](#) ([curved25519_key](#) pk, const [curved25519_key](#) e)

5.321.1 Typedef Documentation

5.321.1.1 typedef unsigned char curved25519_key[32]

Definition at line 10 of file ed25519-donna.h.

5.321.1.2 typedef unsigned char ed25519_public_key[32]

Definition at line 7 of file ed25519-donna.h.

5.321.1.3 typedef unsigned char ed25519_secret_key[32]

Definition at line 8 of file ed25519-donna.h.

5.321.1.4 typedef unsigned char ed25519_signature[64]

Definition at line 6 of file ed25519-donna.h.

5.321.2 Function Documentation

5.321.2.1 void curved25519_scalarmult_basepoint (curved25519_key *pk*, const curved25519_key *e*)

Definition at line 115 of file ed25519.c.

References ALIGN, ge25519_t::y, and ge25519_t::z.

Referenced by main().

5.321.2.2 void curved25519_scalarmult_basepoint_sse2 (curved25519_key *pk*, const curved25519_key *e*)

Referenced by main().

5.321.2.3 void ed25519_publickey (const ed25519_secret_key *sk*, ed25519_public_key *pk*)

Definition at line 35 of file ed25519.c.

References A, and ALIGN.

Referenced by main().

5.321.2.4 void ed25519_publickey_sse2 (const ed25519_secret_key *sk*, ed25519_public_key *pk*)

Referenced by main().

5.321.2.5 void ed25519_randombytes_unsafe (void * *out*, size_t *count*)

Definition at line 88 of file ed25519-randombytes.h.

References RAND_bytes_d.

Referenced by ed25519_sign_open_batch().

5.321.2.6 void ed25519_randombytes_unsafe_sse2 (void * *out*, size_t *count*)

5.321.2.7 void ed25519_sign (const unsigned char * *m*, size_t *mlen*, const ed25519_secret_key *sk*, const ed25519_public_key *pk*, ed25519_signature *RS*)

Definition at line 49 of file ed25519.c.

References ALIGN.

Referenced by encrypted_chunk_set(), main(), org_sigmet_generate(), signature_full_get(), signature_tree_get(), user_request_generate(), user_request_rotation(), and user_request_sign().

5.321.2.8 int ed25519_sign_open (const unsigned char * *m*, size_t *mlen*, const ed25519_public_key *pk*, const ed25519_signature *RS*)

Definition at line 84 of file ed25519.c.

References A, ALIGN, and ed25519_verify().

Referenced by ed25519_sign_open_batch(), and main().

5.321.2.9 `int ed25519_sign_open_batch (const unsigned char ** m, size_t * mlen, const unsigned char ** pk, const unsigned char ** RS, size_t num, int * valid)`

Definition at line 205 of file ed25519-donna-batchverify.h.

References ALIGN, ED25519_FN, ed25519_randombytes_unsafe(), ed25519_sign_open(), max_batch_size, batch_heap_t::points, batch_heap_t::r, and batch_heap_t::scalars.

5.321.2.10 `int ed25519_sign_open_batch_sse2 (const unsigned char ** m, size_t * mlen, const unsigned char ** pk, const unsigned char ** RS, size_t num, int * valid)`

5.321.2.11 `int ed25519_sign_open_sse2 (const unsigned char * m, size_t mlen, const ed25519_public_key pk, const ed25519_signature RS)`

Referenced by main().

5.321.2.12 `void ed25519_sign_sse2 (const unsigned char * m, size_t mlen, const ed25519_secret_key sk, const ed25519_public_key pk, ed25519_signature RS)`

Referenced by main().

5.322 src/providers/dime/ed25519/ed25519-hash-custom.h File Reference

5.323 src/providers/dime/ed25519/ed25519-hash.h File Reference

```
#include <openssl/sha.h>
#include "providers/symbols.h"
```

Typedefs

- typedef SHA512_CTX [ed25519_hash_context](#)

5.323.1 Typedef Documentation

5.323.1.1 typedef SHA512_CTX ed25519_hash_context

Definition at line 197 of file ed25519-hash.h.

5.324 src/providers/dime/ed25519/ed25519-randombytes-custom.h File Reference

5.325 src/providers/dime/ed25519/ed25519-randombytes.h File Reference

```
#include <openssl/rand.h>
#include "symbols.h"
```

Functions

- void ED25519_FN() [ed25519_randombytes_unsafe](#) (void *p, size_t len)

5.325.1 Function Documentation

5.325.1.1 void ED25519_FN() [ed25519_randombytes_unsafe](#) (void *p, size_t len)

Definition at line 88 of file ed25519-randombytes.h.

References [RAND_bytes_d](#).

Referenced by [ed25519_sign_open_batch](#)().

5.326 src/providers/dime/ed25519/ed25519.c File Reference

```
#include "ed25519-donna.h"
#include "ed25519.h"
#include "ed25519-randombytes.h"
#include "ed25519-hash.h"
#include "../ed25519/ed25519-donna-batchverify.h"
```

Functions

- void ED25519_FN() [ed25519_publickey](#) (const [ed25519_secret_key](#) sk, [ed25519_public_key](#) pk)
- void ED25519_FN() [ed25519_sign](#) (const unsigned char *m, size_t mlen, const [ed25519_secret_key](#) sk, const [ed25519_public_key](#) pk, [ed25519_signature](#) RS)
- int ED25519_FN() [ed25519_sign_open](#) (const unsigned char *m, size_t mlen, const [ed25519_public_key](#) pk, const [ed25519_signature](#) RS)
- void ED25519_FN() [curved25519_scalarmult_basepoint](#) ([curved25519_key](#) pk, const [curved25519_key](#) e)

5.326.1 Function Documentation

5.326.1.1 void ED25519_FN() curved25519_scalarmult_basepoint (curved25519_key pk, const curved25519_key e)

Definition at line 115 of file ed25519.c.

References [ALIGN](#), [ge25519_t::y](#), and [ge25519_t::z](#).

Referenced by [main\(\)](#).

5.326.1.2 void ED25519_FN() ed25519_publickey (const ed25519_secret_key sk, ed25519_public_key pk)

Definition at line 35 of file ed25519.c.

References [A](#), and [ALIGN](#).

Referenced by [main\(\)](#).

5.326.1.3 void ED25519_FN() ed25519_sign (const unsigned char * m, size_t mlen, const ed25519_secret_key sk, const ed25519_public_key pk, ed25519_signature RS)

Definition at line 49 of file ed25519.c.

References [ALIGN](#).

Referenced by [encrypted_chunk_set\(\)](#), [main\(\)](#), [org_sigmet_generate\(\)](#), [signature_full_get\(\)](#), [signature_tree_get\(\)](#), [user_request_generate\(\)](#), [user_request_rotation\(\)](#), and [user_request_sign\(\)](#).

5.326.1.4 int ED25519_FN() ed25519_sign_open (const unsigned char * m, size_t mlen, const ed25519_public_key pk, const ed25519_signature RS)

Definition at line 84 of file ed25519.c.

References [A](#), [ALIGN](#), and [ed25519_verify\(\)](#).

Referenced by [ed25519_sign_open_batch\(\)](#), and [main\(\)](#).

5.327 src/providers/prime/cryptography/ed25519.c File Reference

```
#include "magma.h"
#include <openssl/curve25519.h>
```

Functions

- [ed25519_key_type_t ed25519_type \(ed25519_key_t *key\)](#)
 - [void ed25519_free \(ed25519_key_t *key\)](#)
 - [ed25519_key_t * ed25519_alloc \(void\)](#)
- ed25519.c*
- [ed25519_key_t * ed25519_generate \(void\)](#)
 - [stringer_t * ed25519_public_get \(ed25519_key_t *key, stringer_t *output\)](#)
 - [stringer_t * ed25519_private_get \(ed25519_key_t *key, stringer_t *output\)](#)
 - [ed25519_key_t * ed25519_public_set \(stringer_t *key\)](#)
 - [ed25519_key_t * ed25519_private_set \(stringer_t *key\)](#)
 - [stringer_t * ed25519_sign \(ed25519_key_t *key, stringer_t *data, stringer_t *output\)](#)
 - [int_t ed25519_verify \(ed25519_key_t *key, stringer_t *data, stringer_t *signature\)](#)

Verify an Ed25519 signature using the EdDSA algorithm.

5.327.1 Function Documentation

5.327.1.1 ed25519_key_t* ed25519_alloc (void)

ed25519.c

Definition at line 34 of file ed25519.c.

References [ed25519_key_t](#), [log_pedantic](#), [mm_alloc\(\)](#), and [mm_wipe\(\)](#).

Referenced by [ed25519_generate\(\)](#), [ed25519_private_set\(\)](#), and [ed25519_public_set\(\)](#).

5.327.1.2 void ed25519_free (ed25519_key_t * key)

Definition at line 19 of file ed25519.c.

References [log_pedantic](#), and [mm_free\(\)](#).

Referenced by [encrypted_message_free\(\)](#), [ephemeral_chunk_free\(\)](#), [org_key_free\(\)](#), [org_signet_free\(\)](#), [user_key_free\(\)](#), and [user_signet_free\(\)](#).

5.327.1.3 ed25519_key_t* ed25519_generate (void)

Definition at line 48 of file ed25519.c.

References [ed25519_alloc\(\)](#), [ED25519_KEY_PRIV_LEN](#), [ED25519_KEY_PUB_LEN](#), [ed25519_key_t](#), [ED25519_keypair_d](#), [ED25519_PRIV](#), and [mm_copy\(\)](#).

Referenced by [naked_message_set\(\)](#), [org_key_generate\(\)](#), and [user_key_generate\(\)](#).

5.327.1.4 stringer_t* ed25519_private_get (ed25519_key_t * key, stringer_t * output)

Definition at line 90 of file ed25519.c.

References ED25519_KEY_PRIV_LEN, ED25519_PRIV, log_pedantic, mm_copy(), st_alloc(), st_avail_get(), st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and st_wipe().

Referenced by org_key_get(), and user_key_get().

5.327.1.5 ed25519_key_t* ed25519_private_set (stringer_t * key)

Definition at line 132 of file ed25519.c.

References ed25519_alloc(), ED25519_KEY_PRIV_LEN, ed25519_key_t, ED25519_keypair_from_seed_d, ED25519_PRIV, log_info, mm_copy(), st_data_get(), st_empty, and st_length_get().

Referenced by org_key_set(), and user_key_set().

5.327.1.6 stringer_t* ed25519_public_get (ed25519_key_t * key, stringer_t * output)

Definition at line 65 of file ed25519.c.

References ED25519_KEY_PUB_LEN, ED25519_PRIV, ED25519_PUB, log_pedantic, mm_copy(), st_alloc(), st_avail_get(), st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and st_wipe().

Referenced by ephemeral_chunk_get(), org_signet_fingerprint(), org_signet_generate(), org_signet_get(), org_signet_verify(), user_request_generate(), user_request_get(), user_request_rotation(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_get(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.327.1.7 ed25519_key_t* ed25519_public_set (stringer_t * key)

Definition at line 114 of file ed25519.c.

References ed25519_alloc(), ED25519_KEY_PUB_LEN, ed25519_key_t, ED25519_PUB, log_info, mm_copy(), st_data_get(), st_empty, and st_length_get().

Referenced by ephemeral_chunk_get(), ephemeral_chunk_set(), org_signet_generate(), org_signet_set(), user_request_generate(), user_request_rotation(), user_request_set(), user_request_sign(), and user_signet_set().

5.327.1.8 stringer_t* ed25519_sign (ed25519_key_t * key, stringer_t * data, stringer_t * output)

Definition at line 151 of file ed25519.c.

References ED25519_PRIV, ED25519_sign_d, ED25519_SIGNATURE_LEN, log_info, log_pedantic, MEMORYBUF, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty, st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), and st_valid_tracked().

5.327.1.9 ed25519_key_type_t ed25519_type (ed25519_key_t * key)

Definition at line 11 of file ed25519.c.

References ED25519_ERR.

Referenced by encrypted_chunk_get(), encrypted_chunk_set(), part_decrypt(), part_encrypt(), signature_full_get(), signature_full_verify(), signature_tree_get(), and signature_tree_verify().

5.327.1.10 int_t ed25519_verify (ed25519_key_t * *key*, stringer_t * *data*, stringer_t * *signature*)

Verify an Ed25519 signature using the EdDSA algorithm.

Parameters:

- key* The public signing key.
- data* The data being verified.
- signature* The signature.

Returns:

- 0 for successful signature verification, -1 for a signature verification failures, -2 for processing or parameter issue.

Definition at line 196 of file ed25519.c.

References ED25519_PRIV, ED25519_PUB, ED25519_SIGNATURE_LEN, ed25519_verify(), ED25519_verify_d, log_info, log_pedantic, MEMORYBUF, ssl_error_string(), st_data_get(), st_empty, and st_length_get().

Referenced by ed25519_sign_open(), ed25519_verify(), encrypted_chunk_get(), org_signet_verify(), signature_full_verify(), signature_tree_verify(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.328 src/providers/dime/ed25519/ed25519.h File Reference

```
#include <stdlib.h>
```

Defines

- `#define ED25519_SUFFIX _donna`
- `#define ED25519_FN3(fn, suffix) fn##suffix`
- `#define ED25519_FN2(fn, suffix) ED25519_FN3(fn,suffix)`
- `#define ED25519_FN(fn) ED25519_FN2(fn,ED25519_SUFFIX)`

Typedefs

- `typedef unsigned char ed25519_signature [64]`
- `typedef unsigned char ed25519_public_key [32]`
- `typedef unsigned char ed25519_secret_key [32]`
- `typedef unsigned char curved25519_key [32]`

Functions

- `void ed25519_publickey_donna (const ed25519_secret_key sk, ed25519_public_key pk)`
- `int ed25519_sign_open_donna (const unsigned char *m, size_t mlen, const ed25519_public_key pk, const ed25519_signature RS)`
- `void ed25519_sign_donna (const unsigned char *m, size_t mlen, const ed25519_secret_key sk, const ed25519_public_key pk, ed25519_signature RS)`
- `int ed25519_sign_open_batch_donna (const unsigned char **m, size_t *mlen, const unsigned char **pk, const unsigned char **RS, size_t num, int *valid)`
- `void ed25519_randombytes_unsafe_donna (void *out, size_t count)`
- `void curved25519_scalarmult_basepoint_donna (curved25519_key pk, const curved25519_key e)`

5.328.1 Define Documentation

5.328.1.1 `#define ED25519_FN(fn) ED25519_FN2(fn,ED25519_SUFFIX)`

Definition at line 13 of file ed25519.h.

Referenced by ed25519_sign_open_batch().

5.328.1.2 `#define ED25519_FN2(fn, suffix) ED25519_FN3(fn,suffix)`

Definition at line 12 of file ed25519.h.

5.328.1.3 `#define ED25519_FN3(fn, suffix) fn##suffix`

Definition at line 11 of file ed25519.h.

5.328.1.4 `#define ED25519_SUFFIX _donna`

Definition at line 10 of file ed25519.h.

5.328.2 Typedef Documentation

5.328.2.1 `typedef unsigned char curved25519_key[32]`

Definition at line 19 of file ed25519.h.

5.328.2.2 `typedef unsigned char ed25519_public_key[32]`

Definition at line 16 of file ed25519.h.

5.328.2.3 `typedef unsigned char ed25519_secret_key[32]`

Definition at line 17 of file ed25519.h.

5.328.2.4 `typedef unsigned char ed25519_signature[64]`

Definition at line 15 of file ed25519.h.

5.328.3 Function Documentation

5.328.3.1 `void curved25519_scalarmult_basepoint_donna (curved25519_key pk, const curved25519_key e)`

5.328.3.2 `void ed25519_publickey_donna (const ed25519_secret_key sk, ed25519_public_key pk)`

Referenced by `_deserialize_ed25519_privkey()`, `_generate_ed25519_keypair()`, and `_load_ed25519_privkey()`.

5.328.3.3 `void ed25519_randombytes_unsafe_donna (void * out, size_t count)`

5.328.3.4 `void ed25519_sign_donna (const unsigned char * m, size_t mlen, const ed25519_secret_key sk, const ed25519_public_key pk, ed25519_signature RS)`

Referenced by `_ed25519_sign_data()`.

5.328.3.5 `int ed25519_sign_open_batch_donna (const unsigned char ** m, size_t * mlen, const unsigned char ** pk, const unsigned char ** RS, size_t num, int * valid)`

5.328.3.6 `int ed25519_sign_open_donna (const unsigned char * m, size_t mlen, const ed25519_public_key pk, const ed25519_signature RS)`

Referenced by `_ed25519_verify_sig()`.

5.329 src/providers/dime/ed25519/fuzz/curve25519-ref10.c File Reference

```
#include <stdint.h>
```

Typedefs

- typedef int32_t [crypto_int32](#)
- typedef int64_t [crypto_int64](#)
- typedef uint64_t [crypto_uint64](#)
- typedef [crypto_int32](#) [fe](#) [10]

Functions

- void [fe_0](#) ([fe](#) h)
- void [fe_1](#) ([fe](#) h)
- void [fe_add](#) ([fe](#) h, [fe](#) f, [fe](#) g)
- void [fe_copy](#) ([fe](#) h, [fe](#) f)
- void [fe_cswap](#) ([fe](#) f, [fe](#) g, unsigned int b)
- void [fe_frombytes](#) ([fe](#) h, const unsigned char *s)
- void [fe_mul](#) ([fe](#) h, [fe](#) f, [fe](#) g)
- void [fe_mull121666](#) ([fe](#) h, [fe](#) f)
- void [fe_sq](#) ([fe](#) h, [fe](#) f)
- void [fe_sub](#) ([fe](#) h, [fe](#) f, [fe](#) g)
- void [fe_tobytes](#) (unsigned char *s, [fe](#) h)
- void [fe_invert](#) ([fe](#) out, [fe](#) z)
- int [crypto_scalarmult_ref10](#) (unsigned char *q, const unsigned char *n, const unsigned char *p)
- int [crypto_scalarmult_base_ref10](#) (unsigned char *q, const unsigned char *n)

5.329.1 Typedef Documentation

5.329.1.1 typedef int32_t crypto_int32

Definition at line 3 of file curve25519-ref10.c.

5.329.1.2 typedef int64_t crypto_int64

Definition at line 4 of file curve25519-ref10.c.

5.329.1.3 typedef uint64_t crypto_uint64

Definition at line 5 of file curve25519-ref10.c.

5.329.1.4 typedef crypto_int32 fe[10]

Definition at line 7 of file curve25519-ref10.c.

5.329.2 Function Documentation

5.329.2.1 `int crypto_scalarmult_base_ref10 (unsigned char * q, const unsigned char * n)`

Definition at line 1268 of file curve25519-ref10.c.

References `crypto_scalarmult_ref10()`.

Referenced by `main()`.

5.329.2.2 `int crypto_scalarmult_ref10 (unsigned char * q, const unsigned char * n, const unsigned char * p)`

Definition at line 1082 of file curve25519-ref10.c.

References `fe_0()`, `fe_1()`, `fe_add()`, `fe_copy()`, `fe_cswap()`, `fe_frombytes()`, `fe_invert()`, `fe_mul()`, `fe_mul121666()`, `fe_sq()`, `fe_sub()`, and `fe_tobytes()`.

Referenced by `crypto_scalarmult_base_ref10()`.

5.329.2.3 `void fe_0 (fe h)`

Definition at line 13 of file curve25519-ref10.c.

Referenced by `crypto_scalarmult_ref10()`.

5.329.2.4 `void fe_1 (fe h)`

Definition at line 31 of file curve25519-ref10.c.

Referenced by `crypto_scalarmult_ref10()`.

5.329.2.5 `void fe_add (fe h, fe f, fe g)`

Definition at line 57 of file curve25519-ref10.c.

Referenced by `crypto_scalarmult_ref10()`.

5.329.2.6 `void fe_copy (fe h, fe f)`

Definition at line 105 of file curve25519-ref10.c.

Referenced by `crypto_scalarmult_ref10()`.

5.329.2.7 `void fe_cswap (fe f, fe g, unsigned int b)`

Definition at line 137 of file curve25519-ref10.c.

Referenced by `crypto_scalarmult_ref10()`.

5.329.2.8 `void fe_frombytes (fe h, const unsigned char * s)`

Definition at line 221 of file curve25519-ref10.c.

Referenced by `crypto_scalarmult_ref10()`.

5.329.2.9 void fe_invert (fe *out*, fe *z*)

Definition at line 909 of file curve25519-ref10.c.

References fe_mul(), and fe_sq().

Referenced by crypto_scalarmult_ref10().

5.329.2.10 void fe_mul (fe *h*, fe *f*, fe *g*)

Definition at line 301 of file curve25519-ref10.c.

Referenced by crypto_scalarmult_ref10(), and fe_invert().

5.329.2.11 void fe_mul121666 (fe *h*, fe *f*)

Definition at line 531 of file curve25519-ref10.c.

Referenced by crypto_scalarmult_ref10().

5.329.2.12 void fe_sq (fe *h*, fe *f*)

Definition at line 603 of file curve25519-ref10.c.

Referenced by crypto_scalarmult_ref10(), and fe_invert().

5.329.2.13 void fe_sub (fe *h*, fe *f*, fe *g*)

Definition at line 747 of file curve25519-ref10.c.

Referenced by crypto_scalarmult_ref10().

5.329.2.14 void fe_tobytes (unsigned char * *s*, fe *h*)

Definition at line 816 of file curve25519-ref10.c.

Referenced by crypto_scalarmult_ref10().

5.330 src/providers/dime/ed25519/fuzz/curve25519-ref10.h File Reference

Functions

- int [crypto_scalarmult_base_ref10](#) (unsigned char *q, const unsigned char *n)
- int [crypto_scalarmult_ref10](#) (unsigned char *q, const unsigned char *n, const unsigned char *p)

5.330.1 Function Documentation

5.330.1.1 int [crypto_scalarmult_base_ref10](#) (unsigned char * *q*, const unsigned char * *n*)

Definition at line 1268 of file curve25519-ref10.c.

References [crypto_scalarmult_ref10](#)().

Referenced by [main](#)().

5.330.1.2 int [crypto_scalarmult_ref10](#) (unsigned char * *q*, const unsigned char * *n*, const unsigned char * *p*)

Definition at line 1082 of file curve25519-ref10.c.

References [fe_0](#)(), [fe_1](#)(), [fe_add](#)(), [fe_copy](#)(), [fe_cswap](#)(), [fe_frombytes](#)(), [fe_invert](#)(), [fe_mul](#)(), [fe_mul121666](#)(), [fe_sq](#)(), [fe_sub](#)(), and [fe_tobytes](#)().

Referenced by [crypto_scalarmult_base_ref10](#)().

5.331 src/providers/dime/ed25519/fuzz/ed25519-donna-sse2.c File Reference

```
#include "../..ed25519/ed25519.c"
```

Defines

- #define [ED25519_SUFFIX_sse2](#)
- #define [ED25519_SSE2](#)

5.331.1 Define Documentation

5.331.1.1 #define ED25519_SSE2

Definition at line 3 of file ed25519-donna-sse2.c.

5.331.1.2 #define ED25519_SUFFIX_sse2

Definition at line 2 of file ed25519-donna-sse2.c.

5.332 src/providers/dime/ed25519/fuzz/ed25519-donna.c File Reference

```
#include "../..ed25519/ed25519.c"
```

5.333 src/providers/dime/ed25519/fuzz/ed25519-ref10.c File Reference

```
#include <stdint.h>
#include <stddef.h>
#include <string.h>
#include <openssl/sha.h>
```

Data Structures

- struct [ge_p2](#)
- struct [ge_p3](#)
- struct [ge_p1p1](#)
- struct [ge_precomp](#)
- struct [ge_cached](#)

Defines

- `#define F\(i\) differentbits |= x[i] ^ y[i];`

Typedefs

- typedef int32_t [crypto_int32](#)
- typedef uint32_t [crypto_uint32](#)
- typedef int64_t [crypto_int64](#)
- typedef uint64_t [crypto_uint64](#)
- typedef [crypto_int32](#) fe [10]

Functions

- int [crypto_sign_pk_ref10](#) (unsigned char *pk, unsigned char *sk)
- int [crypto_sign_ref10](#) (unsigned char *sm, unsigned long long *smlen, const unsigned char *m, unsigned long long mlen, const unsigned char *sk)
- int [crypto_sign_open_ref10](#) (unsigned char *m, unsigned long long *mlen, const unsigned char *sm, unsigned long long smlen, const unsigned char *pk)

5.333.1 Define Documentation

- 5.333.1.1 `#define F\(i\) differentbits |= x[i] ^ y[i];`

5.333.2 Typedef Documentation

- 5.333.2.1 typedef int32_t [crypto_int32](#)

Definition at line 245 of file ed25519-ref10.c.

- 5.333.2.2 typedef int64_t [crypto_int64](#)

Definition at line 247 of file ed25519-ref10.c.

5.333.2.3 typedef uint32_t crypto_uint32

Definition at line 246 of file ed25519-ref10.c.

5.333.2.4 typedef uint64_t crypto_uint64

Definition at line 248 of file ed25519-ref10.c.

5.333.2.5 typedef crypto_int32 fe[10]

Definition at line 250 of file ed25519-ref10.c.

5.333.3 Function Documentation**5.333.3.1 int crypto_sign_open_ref10 (unsigned char * *m*, unsigned long long * *mlen*, const unsigned char * *sm*, unsigned long long *smlen*, const unsigned char * *pk*)**

Definition at line 4613 of file ed25519-ref10.c.

Referenced by main().

5.333.3.2 int crypto_sign_pk_ref10 (unsigned char * *pk*, unsigned char * *sk*)

Definition at line 4561 of file ed25519-ref10.c.

Referenced by main().

5.333.3.3 int crypto_sign_ref10 (unsigned char * *sm*, unsigned long long * *smlen*, const unsigned char * *m*, unsigned long long *mlen*, const unsigned char * *sk*)

Definition at line 4579 of file ed25519-ref10.c.

Referenced by main().

5.334 src/providers/dime/ed25519/fuzz/ed25519-ref10.h File Reference

Functions

- int [crypto_sign_pk_ref10](#) (unsigned char *pk, unsigned char *sk)
- int [crypto_sign_ref10](#) (unsigned char *sm, unsigned long long *smlen, const unsigned char *m, unsigned long long mlen, const unsigned char *sk)
- int [crypto_sign_open_ref10](#) (unsigned char *m, unsigned long long *mlen, const unsigned char *sm, unsigned long long smlen, const unsigned char *pk)

5.334.1 Function Documentation

5.334.1.1 int [crypto_sign_open_ref10](#) (unsigned char * *m*, unsigned long long * *mlen*, const unsigned char * *sm*, unsigned long long *smlen*, const unsigned char * *pk*)

Definition at line 4613 of file ed25519-ref10.c.

Referenced by main().

5.334.1.2 int [crypto_sign_pk_ref10](#) (unsigned char * *pk*, unsigned char * *sk*)

Definition at line 4561 of file ed25519-ref10.c.

Referenced by main().

5.334.1.3 int [crypto_sign_ref10](#) (unsigned char * *sm*, unsigned long long * *smlen*, const unsigned char * *m*, unsigned long long *mlen*, const unsigned char * *sk*)

Definition at line 4579 of file ed25519-ref10.c.

Referenced by main().

5.335 src/providers/dime/ed25519/fuzz/fuzz-curve25519.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <stdio.h>
#include "../ed25519/fuzz/ed25519-donna.h"
#include "../ed25519/fuzz/curve25519-ref10.h"
```

Defines

- #define [rotl32](#)(x, k) ((x << k) | (x >> (32 - k)))
- #define [quarter](#)(a, b, c, d)

Functions

- void [prng](#) (unsigned char *out, size_t bytes)
- int [main](#) ()

5.335.1 Define Documentation

5.335.1.1 #define quarter(a, b, c, d)

Value:

```
x[a] += x[b]; t = x[d]^x[a]; x[d] = rotl32(t, 16); \
x[c] += x[d]; t = x[b]^x[c]; x[b] = rotl32(t, 12); \
x[a] += x[b]; t = x[d]^x[a]; x[d] = rotl32(t, 8); \
x[c] += x[d]; t = x[b]^x[c]; x[b] = rotl32(t, 7);
```

Referenced by [prng\(\)](#).

5.335.1.2 #define rotl32(x, k) ((x << k) | (x >> (32 - k)))

5.335.2 Function Documentation

5.335.2.1 int main (void)

Definition at line 119 of file [fuzz-curve25519.c](#).

References [crypto_scalarmult_base_ref10\(\)](#), [curved25519_scalarmult_basepoint\(\)](#), [curved25519_scalarmult_basepoint_sse2\(\)](#), and [prng\(\)](#).

5.335.2.2 void prng (unsigned char * out, size_t bytes)

Definition at line 49 of file [fuzz-curve25519.c](#).

References [quarter](#).

Referenced by [main\(\)](#).

5.336 src/providers/dime/ed25519/fuzz/fuzz-ed25519.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <stdio.h>
#include "../ed25519/fuzz/ed25519-donna.h"
#include "../ed25519/fuzz/ed25519-ref10.h"
```

Data Structures

- struct [random_data_t](#)
- struct [generated_data_t](#)

Defines

- #define [rotl32](#)(x, k) ((x << k) | (x >> (32 - k)))
- #define [quarter](#)(a, b, c, d)

Typedefs

- typedef struct [random_data_t](#) [random_data](#)
- typedef struct [generated_data_t](#) [generated_data](#)

Functions

- void [prng](#) (unsigned char *out, size_t bytes)
- int [main](#) ()

5.336.1 Define Documentation

5.336.1.1 #define quarter(a, b, c, d)

Value:

```
x[a] += x[b]; t = x[d]^x[a]; x[d] = rotl32(t, 16); \
x[c] += x[d]; t = x[b]^x[c]; x[b] = rotl32(t, 12); \
x[a] += x[b]; t = x[d]^x[a]; x[d] = rotl32(t, 8); \
x[c] += x[d]; t = x[b]^x[c]; x[b] = rotl32(t, 7);
```

5.336.1.2 `#define rotl32(x, k) ((x << k) | (x >> (32 - k)))`

5.336.2 Typedef Documentation

5.336.2.1 `typedef struct generated_data_t generated_data`

5.336.2.2 `typedef struct random_data_t random_data`

5.336.3 Function Documentation

5.336.3.1 `int main (void)`

Definition at line 143 of file fuzz-ed25519.c.

References `crypto_sign_open_ref10()`, `crypto_sign_pk_ref10()`, `crypto_sign_ref10()`, `ed25519_publickey()`, `ed25519_publickey_sse2()`, `ed25519_sign()`, `ed25519_sign_open()`, `ed25519_sign_open_sse2()`, `ed25519_sign_sse2()`, `random_data_t::m`, `generated_data_t::pk`, `prng()`, `generated_data_t::sig`, `random_data_t::sk`, and `generated_data_t::valid`.

5.336.3.2 `void prng (unsigned char * out, size_t bytes)`

Definition at line 48 of file fuzz-ed25519.c.

References `quarter`.

5.337 src/providers/dime/ed25519/modm-donna-32bit.h File Reference

Defines

- #define `bignum256modm_bits_per_limb` 30
- #define `bignum256modm_limb_size` 9

Typedefs

- typedef uint32_t `bignum256modm_element_t`
- typedef `bignum256modm_element_t` `bignum256modm` [9]

5.337.1 Define Documentation

5.337.1.1 #define `bignum256modm_bits_per_limb` 30

Definition at line 15 of file `modm-donna-32bit.h`.

5.337.1.2 #define `bignum256modm_limb_size` 9

Definition at line 16 of file `modm-donna-32bit.h`.

5.337.2 Typedef Documentation

5.337.2.1 typedef `bignum256modm_element_t` `bignum256modm`[9]

Definition at line 19 of file `modm-donna-32bit.h`.

5.337.2.2 typedef uint32_t `bignum256modm_element_t`

Definition at line 18 of file `modm-donna-32bit.h`.

5.338 src/providers/dime/ed25519/modm-donna-64bit.h File Reference

Defines

- #define [bignum256modm_bits_per_limb](#) 56
- #define [bignum256modm_limb_size](#) 5

Typedefs

- typedef uint64_t [bignum256modm_element_t](#)
- typedef [bignum256modm_element_t](#) [bignum256modm](#) [5]

5.338.1 Define Documentation

5.338.1.1 #define [bignum256modm_bits_per_limb](#) 56

Definition at line 15 of file modm-donna-64bit.h.

5.338.1.2 #define [bignum256modm_limb_size](#) 5

Definition at line 16 of file modm-donna-64bit.h.

5.338.2 Typedef Documentation

5.338.2.1 typedef [bignum256modm_element_t](#) [bignum256modm](#)[5]

Definition at line 19 of file modm-donna-64bit.h.

5.338.2.2 typedef uint64_t [bignum256modm_element_t](#)

Definition at line 18 of file modm-donna-64bit.h.

5.339 src/providers/dime/ed25519/regression.h File Reference

5.340 src/providers/dime/ed25519/test-internals.c File Reference

```
#include <stdio.h>
#include "../ed25519/ed25519-donna.h"
```

Functions

- int [main](#) ()

5.340.1 Function Documentation

5.340.1.1 int main (void)

Definition at line 164 of file test-internals.c.

5.341 src/providers/dime/ed25519/test-ticks.h File Reference

```
#include "../ed25519/ed25519-donna-portable-identify.h"
```

Defines

- #define [timeit](#)(x, minvar)
- #define [maxticks](#) 0xfffffffffffffffffull

5.341.1 Define Documentation

5.341.1.1 #define maxticks 0xfffffffffffffffffull

Definition at line 49 of file test-ticks.h.

5.341.1.2 #define timeit(x, minvar)

Value:

```
ticks = get_ticks();          \  
    x;                          \  
    ticks = get_ticks() - ticks; \  
    if (ticks < minvar)        \  
        minvar = ticks;
```

Definition at line 42 of file test-ticks.h.

5.342 src/providers/dime/ed25519/test.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "test-ticks.h"
#include "../ed25519/ed25519.h"
#include "../ed25519/test-ticks.h"
#include "../ed25519/regression.h"
```

Data Structures

- struct [test_data_t](#)

Defines

- #define [test_batch_count](#) 64
- #define [test_batch_rounds](#) 96

Typedefs

- typedef struct [test_data_t](#) [test_data](#)
- typedef enum [batch_test_t](#) [batch_test](#)

Enumerations

- enum [batch_test_t](#) { [batch_no_errors](#) = 0, [batch_wrong_message](#) = 1, [batch_wrong_pk](#) = 2, [batch_wrong_sig](#) = 3 }

Functions

- int [main](#) (void)

Variables

- [test_data](#) dataset []
- const [curved25519_key](#) [curved25519_expected](#)
- unsigned char [batch_point_buffer](#) [3][32]

5.342.1 Define Documentation

5.342.1.1 #define test_batch_count 64

Definition at line 111 of file test.c.

5.342.1.2 #define test_batch_rounds 96

Definition at line 112 of file test.c.

5.342.2 Typedef Documentation

5.342.2.1 typedef enum batch_test_t batch_test

5.342.2.2 typedef struct test_data_t test_data

5.342.3 Enumeration Type Documentation

5.342.3.1 enum batch_test_t

Enumerator:

batch_no_errors
batch_wrong_message
batch_wrong_pk
batch_wrong_sig

Definition at line 114 of file test.c.

5.342.4 Function Documentation

5.342.4.1 int main (void)

Definition at line 256 of file test.c.

5.342.5 Variable Documentation

5.342.5.1 unsigned char batch_point_buffer[3][32]

Definition at line 191 of file ed25519-donna-batchverify.h.

5.342.5.2 const curved25519_key curved25519_expected

Initial value:

```
{  
    0xac, 0xce, 0x24, 0xb1, 0xd4, 0xa2, 0x36, 0x21,  
    0x15, 0xe2, 0x3e, 0x84, 0x3c, 0x23, 0x2b, 0x5f,  
    0x95, 0x6c, 0xc0, 0x7b, 0x95, 0x82, 0xd7, 0x93,  
    0xd5, 0x19, 0xb6, 0xf1, 0xfb, 0x96, 0xd6, 0x04  
}
```

Definition at line 61 of file test.c.

5.342.5.3 test_data dataset[]

Definition at line 56 of file test.c.

5.343 src/providers/dime/error_codes.c File Reference

```
#include "dime/error_codes.h"
```

Variables

- `derror_t` const *const `ERR_CRYPT` = &CRYPTO_ERROR
- `derror_t` const *const `ERR_NOMEM` = &NOMEM_ERROR
- `derror_t` const *const `ERR_BAD_PARAM` = &BAD_PARAM_ERROR
- `derror_t` const *const `ERR_FILE_IO` = &FILE_IO_ERROR
- `derror_t` const *const `ERR_ENCODING` = &ENCODING_ERROR

5.343.1 Variable Documentation

5.343.1.1 `derror_t` const* const `ERR_BAD_PARAM` = &BAD_PARAM_ERROR

Definition at line 22 of file `error_codes.c`.

5.343.1.2 `derror_t` const* const `ERR_CRYPT` = &CRYPTO_ERROR

Definition at line 8 of file `error_codes.c`.

Referenced by `encrypt_ctx_new()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, and `encrypt_keypair_generate()`.

5.343.1.3 `derror_t` const* const `ERR_ENCODING` = &ENCODING_ERROR

Definition at line 36 of file `error_codes.c`.

Referenced by `libdime_base64_decode()`.

5.343.1.4 `derror_t` const* const `ERR_FILE_IO` = &FILE_IO_ERROR

Definition at line 29 of file `error_codes.c`.

5.343.1.5 `derror_t` const* const `ERR_NOMEM` = &NOMEM_ERROR

Definition at line 15 of file `error_codes.c`.

5.344 src/providers/dime/error_codes.h File Reference

Data Structures

- struct [derror_t](#)

Enumerations

- enum [dime_errcode_t](#) {
 [ERRCODE_NOMEM](#), [ERRCODE_CRYPT0](#), [ERRCODE_BAD_PARAM](#), [ERRCODE_FILE_IO](#),
 [ERRCODE_ENCODING](#) }

Variables

- [derror_t](#) const *const [ERR_CRYPT0](#)
- [derror_t](#) const *const [ERR_NOMEM](#)
- [derror_t](#) const *const [ERR_BAD_PARAM](#)
- [derror_t](#) const *const [ERR_FILE_IO](#)
- [derror_t](#) const *const [ERR_ENCODING](#)

5.344.1 Enumeration Type Documentation

5.344.1.1 enum [dime_errcode_t](#)

Enumerator:

[ERRCODE_NOMEM](#)
[ERRCODE_CRYPT0](#)
[ERRCODE_BAD_PARAM](#)
[ERRCODE_FILE_IO](#)
[ERRCODE_ENCODING](#)

Definition at line 4 of file [error_codes.h](#).

5.344.2 Variable Documentation

5.344.2.1 [derror_t](#) const* const [ERR_BAD_PARAM](#)

Definition at line 22 of file [error_codes.c](#).

5.344.2.2 [derror_t](#) const* const [ERR_CRYPT0](#)

Definition at line 8 of file [error_codes.c](#).

Referenced by [encrypt_ctx_new\(\)](#), [encrypt_deserialize_privkey\(\)](#), [encrypt_deserialize_pubkey\(\)](#), and [encrypt_keypair_generate\(\)](#).

5.344.2.3 [derror_t](#) const* const [ERR_ENCODING](#)

Definition at line 36 of file [error_codes.c](#).

Referenced by [libdime_base64_decode\(\)](#).

5.344.2.4 derror_t const* const ERR_FILE_IO

Definition at line 29 of file error_codes.c.

5.344.2.5 derror_t const* const ERR_NOMEM

Definition at line 15 of file error_codes.c.

5.345 src/providers/dime/sds/sds.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>
#include "sds.h"
#include "sdsalloc.h"
```

Defines

- #define [SDS_LLSTR_SIZE](#) 21

Functions

- [sds sdsnewlen](#) (const void *init, size_t initlen)
- [sds sdsempty](#) (void)
- [sds sdsnew](#) (const char *init)
- [sds sdsdup](#) (const [sds](#) s)
- void [sdsfree](#) ([sds](#) s)
- void [sdsupdatelen](#) ([sds](#) s)
- void [sdsclear](#) ([sds](#) s)
- [sds sdsMakeRoomFor](#) ([sds](#) s, size_t addlen)
- [sds sdsRemoveFreeSpace](#) ([sds](#) s)
- size_t [sdsAllocSize](#) ([sds](#) s)
- void * [sdsAllocPtr](#) ([sds](#) s)
- void [sdsIncrLen](#) ([sds](#) s, int incr)
- [sds sdsgrowzero](#) ([sds](#) s, size_t len)
- [sds sdscatlen](#) ([sds](#) s, const void *t, size_t len)
- [sds sdscat](#) ([sds](#) s, const char *t)
- [sds sdscatsds](#) ([sds](#) s, const [sds](#) t)
- [sds sdscpylen](#) ([sds](#) s, const char *t, size_t len)
- [sds sdscpy](#) ([sds](#) s, const char *t)
- int [sdsll2str](#) (char *s, long long value)
- int [sdsull2str](#) (char *s, unsigned long long v)
- [sds sdsfromlonglong](#) (long long value)
- [sds sdscatvprintf](#) ([sds](#) s, const char *fmt, va_list ap)
- [sds sdscatprintf](#) ([sds](#) s, const char *fmt,...)
- [sds sdscatfmt](#) ([sds](#) s, char const *fmt,...)
- [sds sdstrim](#) ([sds](#) s, const char *cset)
- void [sdsrange](#) ([sds](#) s, int start, int end)
- void [sdstolower](#) ([sds](#) s)
- void [sdstoupper](#) ([sds](#) s)
- int [sdscmp](#) (const [sds](#) s1, const [sds](#) s2)
- [sds](#) * [sdssplitlen](#) (const char *s, int len, const char *sep, int seplen, int *count)
- void [sdsfreesplitres](#) ([sds](#) *tokens, int count)
- [sds sdscatrepr](#) ([sds](#) s, const char *p, size_t len)
- int [is_hex_digit](#) (char c)
- int [hex_digit_to_int](#) (char c)

- [sds * sdssplitargs](#) (const char *line, int *argc)
- [sds sdsmapchars](#) (sds s, const char *from, const char *to, size_t setlen)
- [sds sdsjoin](#) (char **argv, int argc, char *sep)
- [sds sdsjoinsds](#) (sds *argv, int argc, const char *sep, size_t seplen)

5.345.1 Define Documentation

5.345.1.1 #define SDS_LLSTR_SIZE 21

Definition at line 428 of file sds.c.

Referenced by sdscatfmt(), and sdsfromlonglong().

5.345.2 Function Documentation

5.345.2.1 int hex_digit_to_int (char c)

Definition at line 892 of file sds.c.

Referenced by sdssplitargs().

5.345.2.2 int is_hex_digit (char c)

Definition at line 885 of file sds.c.

Referenced by sdssplitargs().

5.345.2.3 void* sdsAllocPtr (sds s)

Definition at line 285 of file sds.c.

5.345.2.4 size_t sdsAllocSize (sds s)

Definition at line 278 of file sds.c.

5.345.2.5 sds sdscat (sds s, const char * t)

Definition at line 391 of file sds.c.

References sdscatlen().

Referenced by sdscatvprintf(), and sdsjoin().

5.345.2.6 sds sdscatfmt (sds s, char const *fmt, ...)

Definition at line 579 of file sds.c.

References SDS_LLSTR_SIZE, sdsll2str(), sdsMakeRoomFor(), and sdsull2str().

5.345.2.7 sds sdscatlen (sds s, const void *t, size_t len)

Definition at line 376 of file sds.c.

References sdsMakeRoomFor().

Referenced by sdscat(), sdscatrepr(), sdscatsds(), sdsjoinsds(), and sdssplitargs().

5.345.2.8 sds sdscatprintf (sds *s*, const char **fmt*, ...)

Definition at line 554 of file sds.c.

References sdscatvprintf().

Referenced by sdscatrepr().

5.345.2.9 sds sdscatrepr (sds *s*, const char **p*, size_t *len*)

Definition at line 858 of file sds.c.

References sdscatlen(), and sdscatprintf().

5.345.2.10 sds sdscatsds (sds *s*, const sds *t*)

Definition at line 399 of file sds.c.

References sdscatlen().

Referenced by sdsjoinsds().

5.345.2.11 sds sdscatvprintf (sds *s*, const char **fmt*, va_list *ap*)

Definition at line 501 of file sds.c.

References buflen, s_free, s_malloc, and sdscat().

Referenced by sdscatprintf().

5.345.2.12 void sdsclear (sds *s*)

Definition at line 183 of file sds.c.

5.345.2.13 int sdscmp (const sds *s1*, const sds *s2*)

Definition at line 767 of file sds.c.

5.345.2.14 sds sdscopy (sds *s*, const char **t*)

Definition at line 418 of file sds.c.

References sdscopylen().

5.345.2.15 sds sdscopylen (sds *s*, const char **t*, size_t *len*)

Definition at line 405 of file sds.c.

References sdsMakeRoomFor().

Referenced by sdscopy().

5.345.2.16 `sds sdsdup (const sds s)`

Definition at line 150 of file sds.c.

References sdsnewlen().

Referenced by dime_dmtplib_command_ahlo(), dime_dmtplib_command_helo(), dime_dmtplib_command_hist(), dime_dmtplib_command_mail(), dime_dmtplib_command_noop(), dime_dmtplib_command_rcpt(), dime_dmtplib_command_sgnt_domain(), dime_dmtplib_command_sgnt_user(), dime_dmtplib_command_starttls(), dime_dmtplib_command_vrfy_domain(), and dime_dmtplib_command_vrfy_user().

5.345.2.17 `sds sdsempty (void)`

Definition at line 139 of file sds.c.

References sdsnewlen().

Referenced by sdsjoin(), sdsjoinsds(), and sdssplitargs().

5.345.2.18 `void sdsfree (sds s)`

Definition at line 155 of file sds.c.

References s_free.

Referenced by libdime_base64_encode(), sdsfreesplitres(), sdssplitargs(), and sdssplitlen().

5.345.2.19 `void sdsfreesplitres (sds * tokens, int count)`

Definition at line 845 of file sds.c.

References s_free, and sdsfree().

5.345.2.20 `sds sdsfromlonglong (long long value)`

Definition at line 493 of file sds.c.

References SDS_LLSTR_SIZE, sdsll2str(), and sdsnewlen().

5.345.2.21 `sds sdsgrowzero (sds s, size_t len)`

Definition at line 358 of file sds.c.

References sdsMakeRoomFor().

5.345.2.22 `void sdsIncrLen (sds s, int incr)`

Definition at line 312 of file sds.c.

References SDS_HDR_VAR, SDS_TYPE_16, SDS_TYPE_32, SDS_TYPE_5, SDS_TYPE_5_LEN, SDS_TYPE_64, SDS_TYPE_8, SDS_TYPE_BITS, and SDS_TYPE_MASK.

5.345.2.23 `sds sdsjoin (char ** argv, int argc, char * sep)`

Definition at line 1068 of file sds.c.

References sdscat(), and sdsempty().

5.345.2.24 sds sdsjoinsds (sds * *argv*, int *argc*, const char * *sep*, size_t *seplen*)

Definition at line 1080 of file sds.c.

References sdsctlen(), sdsctsd(), and sdsempty().

5.345.2.25 int sdsll2str (char * *s*, long long *value*)

Definition at line 429 of file sds.c.

Referenced by sdsctfmt(), and sdsfromlonglong().

5.345.2.26 sds sdsMakeRoomFor (sds *s*, size_t *addlen*)

Definition at line 194 of file sds.c.

References s_free, s_malloc, s_realloc, SDS_MAX_PREALLOC, SDS_TYPE_5, SDS_TYPE_8, and SDS_TYPE_MASK.

Referenced by sdsctfmt(), sdsctlen(), sdscpylen(), and sdsgrowzero().

5.345.2.27 sds sdsmapchars (sds *s*, const char * *from*, const char * *to*, size_t *setlen*)

Definition at line 1052 of file sds.c.

5.345.2.28 sds sdsnew (const char * *init*)

Definition at line 144 of file sds.c.

References sdsnewlen().

5.345.2.29 sds sdsnewlen (const void * *init*, size_t *initlen*)

Definition at line 81 of file sds.c.

References s_malloc, SDS_HDR_VAR, SDS_TYPE_16, SDS_TYPE_32, SDS_TYPE_5, SDS_TYPE_64, SDS_TYPE_8, and SDS_TYPE_BITS.

Referenced by dime_dntp_command_starttls(), libdime_base64_encode(), sdsdup(), sdsempty(), sdsfromlonglong(), sdsnew(), and sdssplitlen().

5.345.2.30 void sdsrange (sds *s*, int *start*, int *end*)

Definition at line 714 of file sds.c.

5.345.2.31 sds sdsRemoveFreeSpace (sds *s*)

Definition at line 245 of file sds.c.

References s_free, s_malloc, s_realloc, and SDS_TYPE_MASK.

5.345.2.32 sds* sdssplitargs (const char * *line*, int * *argc*)

Definition at line 933 of file sds.c.

References hex_digit_to_int(), is_hex_digit(), s_free, s_malloc, s_realloc, sdsctlen(), sdsempty(), and sdsfree().

5.345.2.33 `sds* sdssplitlen (const char * s, int len, const char * sep, int seplen, int * count)`

Definition at line 795 of file sds.c.

References `s_free`, `s_malloc`, `s_realloc`, `sdsfree()`, and `sdsnewlen()`.

5.345.2.34 `void sdstolower (sds s)`

Definition at line 743 of file sds.c.

5.345.2.35 `void sdstoupper (sds s)`

Definition at line 750 of file sds.c.

5.345.2.36 `sds sdstrim (sds s, const char * cset)`

Definition at line 683 of file sds.c.

5.345.2.37 `int sdsull2str (char * s, unsigned long long v)`

Definition at line 461 of file sds.c.

Referenced by `sdscatfmt()`.

5.345.2.38 `void sdsupdatelen (sds s)`

Definition at line 174 of file sds.c.

5.346 src/providers/dime/sds/sds.h File Reference

```
#include <sys/types.h>
#include <stdarg.h>
#include <stdint.h>
```

Defines

- #define [SDS_MAX_PREALLOC](#) (1024*1024)
- #define [SDS_TYPE_5](#) 0
- #define [SDS_TYPE_8](#) 1
- #define [SDS_TYPE_16](#) 2
- #define [SDS_TYPE_32](#) 3
- #define [SDS_TYPE_64](#) 4
- #define [SDS_TYPE_MASK](#) 7
- #define [SDS_TYPE_BITS](#) 3
- #define [SDS_HDR_VAR](#)(T, s) struct sdshdr##T *sh = (struct sdshdr##T *)((s)-(sizeof(struct sdshdr##T)));
- #define [SDS_HDR](#)(T, s) ((struct sdshdr##T *)((s)-(sizeof(struct sdshdr##T))))
- #define [SDS_TYPE_5_LEN](#)(f) ((f)>>SDS_TYPE_BITS)

Typedefs

- typedef char * [sds](#)

Functions

- struct [__attribute__](#) ((packed)) sdshdr5
- [sds sdsnewlen](#) (const void *init, size_t initlen)
- [sds sdsnew](#) (const char *init)
- [sds sdsempty](#) (void)
- [sds sdsdup](#) (const [sds](#) s)
- void [sdsfree](#) ([sds](#) s)
- [sds sdsgrowzero](#) ([sds](#) s, size_t len)
- [sds sdscatlen](#) ([sds](#) s, const void *t, size_t len)
- [sds sdscat](#) ([sds](#) s, const char *t)
- [sds sdscatsds](#) ([sds](#) s, const [sds](#) t)
- [sds sdscpylen](#) ([sds](#) s, const char *t, size_t len)
- [sds sdscpy](#) ([sds](#) s, const char *t)
- [sds sdscatprintf](#) ([sds](#) s, const char *fmt, va_list ap)
- [sds sdscatprintf](#) ([sds](#) s, const char *fmt,...)
- [sds sdscatfmt](#) ([sds](#) s, char const *fmt,...)
- [sds sdstrim](#) ([sds](#) s, const char *cset)
- void [sdsrange](#) ([sds](#) s, int start, int end)
- void [sdsupdatelen](#) ([sds](#) s)
- void [sdsclear](#) ([sds](#) s)
- int [sdscmp](#) (const [sds](#) s1, const [sds](#) s2)
- [sds * sdssplitlen](#) (const char *s, int len, const char *sep, int seplen, int *count)
- void [sdsfreesplitres](#) ([sds](#) *tokens, int count)
- void [sdstolower](#) ([sds](#) s)
- void [sdstoupper](#) ([sds](#) s)
- [sds sdsfromlonglong](#) (long long value)

- [sds sdscatrepr](#) ([sds](#) s, const char *p, size_t len)
- [sds * sdssplitargs](#) (const char *line, int *argc)
- [sds sdsmapchars](#) ([sds](#) s, const char *from, const char *to, size_t setlen)
- [sds sdsjoin](#) (char **argv, int argc, char *sep)
- [sds sdsjoinsds](#) ([sds](#) *argv, int argc, const char *sep, size_t seplen)
- [sds sdsMakeRoomFor](#) ([sds](#) s, size_t addlen)
- void [sdsIncrLen](#) ([sds](#) s, int incr)
- [sds sdsRemoveFreeSpace](#) ([sds](#) s)
- size_t [sdsAllocSize](#) ([sds](#) s)
- void * [sdsAllocPtr](#) ([sds](#) s)

5.346.1 Define Documentation

5.346.1.1 `#define SDS_HDR(T, s) ((struct sdshdr##T *)((s)-(sizeof(struct sdshdr##T))))`

Definition at line 83 of file sds.h.

5.346.1.2 `#define SDS_HDR_VAR(T, s) struct sdshdr##T *sh = (struct sdshdr##T *)((s)-(sizeof(struct sdshdr##T)));`

Definition at line 82 of file sds.h.

Referenced by [sdsIncrLen\(\)](#), and [sdsnewlen\(\)](#).

5.346.1.3 `#define SDS_MAX_PREALLOC (1024*1024)`

Definition at line 36 of file sds.h.

Referenced by [sdsMakeRoomFor\(\)](#).

5.346.1.4 `#define SDS_TYPE_16 2`

Definition at line 77 of file sds.h.

Referenced by [sdsIncrLen\(\)](#), and [sdsnewlen\(\)](#).

5.346.1.5 `#define SDS_TYPE_32 3`

Definition at line 78 of file sds.h.

Referenced by [sdsIncrLen\(\)](#), and [sdsnewlen\(\)](#).

5.346.1.6 `#define SDS_TYPE_5 0`

Definition at line 75 of file sds.h.

Referenced by [sdsIncrLen\(\)](#), [sdsMakeRoomFor\(\)](#), and [sdsnewlen\(\)](#).

5.346.1.7 `#define SDS_TYPE_5_LEN(f) ((f)>>SDS_TYPE_BITS)`

Definition at line 84 of file sds.h.

Referenced by [sdsIncrLen\(\)](#).

5.346.1.8 `#define SDS_TYPE_64 4`

Definition at line 79 of file sds.h.

Referenced by `sdsIncrLen()`, and `sdsnewlen()`.

5.346.1.9 `#define SDS_TYPE_8 1`

Definition at line 76 of file sds.h.

Referenced by `sdsIncrLen()`, `sdsMakeRoomFor()`, and `sdsnewlen()`.

5.346.1.10 `#define SDS_TYPE_BITS 3`

Definition at line 81 of file sds.h.

Referenced by `sdsIncrLen()`, and `sdsnewlen()`.

5.346.1.11 `#define SDS_TYPE_MASK 7`

Definition at line 80 of file sds.h.

Referenced by `sdsIncrLen()`, `sdsMakeRoomFor()`, and `sdsRemoveFreeSpace()`.

5.346.2 Typedef Documentation

5.346.2.1 `typedef char* sds`

Definition at line 42 of file sds.h.

5.346.3 Function Documentation

5.346.3.1 `struct __attribute__((packed)) [read]`

Definition at line 68 of file sds.h.

5.346.3.2 `void* sdsAllocPtr (sds s)`

Definition at line 285 of file sds.c.

5.346.3.3 `size_t sdsAllocSize (sds s)`

Definition at line 278 of file sds.c.

5.346.3.4 `sds sdscat (sds s, const char * t)`

Definition at line 391 of file sds.c.

References `sdscatlen()`.

Referenced by `sdscatvprintf()`, and `sdsjoin()`.

5.346.3.5 sds sdscatfmt (sds s, char const *fmt, ...)

Definition at line 579 of file sds.c.

References SDS_LLSTR_SIZE, sdsll2str(), sdsMakeRoomFor(), and sdsull2str().

5.346.3.6 sds sdscatlen (sds s, const void *t, size_t len)

Definition at line 376 of file sds.c.

References sdsMakeRoomFor().

Referenced by sdscat(), sdscatrepr(), sdscatsds(), sdsjoinsds(), and sdssplitargs().

5.346.3.7 sds sdscatprintf (sds s, const char *fmt, ...)

Definition at line 554 of file sds.c.

References sdscatvprintf().

Referenced by sdscatrepr().

5.346.3.8 sds sdscatrepr (sds s, const char *p, size_t len)

Definition at line 858 of file sds.c.

References sdscatlen(), and sdscatprintf().

5.346.3.9 sds sdscatsds (sds s, const sds t)

Definition at line 399 of file sds.c.

References sdscatlen().

Referenced by sdsjoinsds().

5.346.3.10 sds sdscatvprintf (sds s, const char *fmt, va_list ap)

Definition at line 501 of file sds.c.

References buflen, s_free, s_malloc, and sdscat().

Referenced by sdscatprintf().

5.346.3.11 void sdsclear (sds s)

Definition at line 183 of file sds.c.

5.346.3.12 int sdscmp (const sds s1, const sds s2)

Definition at line 767 of file sds.c.

5.346.3.13 sds sdscopy (sds s, const char *t)

Definition at line 418 of file sds.c.

References sdscopylen().

5.346.3.14 sds sdscopylen (sds *s*, const char * *t*, size_t *len*)

Definition at line 405 of file sds.c.

References sdsMakeRoomFor().

Referenced by sdscopy().

5.346.3.15 sds sdsdup (const sds *s*)

Definition at line 150 of file sds.c.

References sdsnewlen().

Referenced by dime_dmtplib_command_ahlo(), dime_dmtplib_command_helo(), dime_dmtplib_command_hist(), dime_dmtplib_command_mail(), dime_dmtplib_command_noop(), dime_dmtplib_command_rcpt(), dime_dmtplib_command_sgent_domain(), dime_dmtplib_command_sgent_user(), dime_dmtplib_command_starttls(), dime_dmtplib_command_vrfy_domain(), and dime_dmtplib_command_vrfy_user().

5.346.3.16 sds sdsempty (void)

Definition at line 139 of file sds.c.

References sdsnewlen().

Referenced by sdsjoin(), sdsjoinsds(), and sdssplitargs().

5.346.3.17 void sdsfree (sds *s*)

Definition at line 155 of file sds.c.

References s_free.

Referenced by libdime_base64_encode(), sdsfreesplitres(), sdssplitargs(), and sdssplitlen().

5.346.3.18 void sdsfreesplitres (sds * *tokens*, int *count*)

Definition at line 845 of file sds.c.

References s_free, and sdsfree().

5.346.3.19 sds sdsfromlonglong (long long *value*)

Definition at line 493 of file sds.c.

References SDS_LLSTR_SIZE, sdsll2str(), and sdsnewlen().

5.346.3.20 sds sdsgrowzero (sds *s*, size_t *len*)

Definition at line 358 of file sds.c.

References sdsMakeRoomFor().

5.346.3.21 void sdsIncrLen (sds *s*, int *incr*)

Definition at line 312 of file sds.c.

References SDS_HDR_VAR, SDS_TYPE_16, SDS_TYPE_32, SDS_TYPE_5, SDS_TYPE_5_LEN, SDS_TYPE_64, SDS_TYPE_8, SDS_TYPE_BITS, and SDS_TYPE_MASK.

5.346.3.22 sds sdsjoin (char ** *argv*, int *argc*, char * *sep*)

Definition at line 1068 of file sds.c.

References sdscat(), and sdsempty().

5.346.3.23 sds sdsjoinsds (sds * *argv*, int *argc*, const char * *sep*, size_t *seplen*)

Definition at line 1080 of file sds.c.

References sdscatlen(), sdscatsds(), and sdsempty().

5.346.3.24 sds sdsMakeRoomFor (sds *s*, size_t *addlen*)

Definition at line 194 of file sds.c.

References s_free, s_malloc, s_realloc, SDS_MAX_PREALLOC, SDS_TYPE_5, SDS_TYPE_8, and SDS_TYPE_MASK.

Referenced by sdscatfmt(), sdscatlen(), sdscpylen(), and sdsgrowzero().

5.346.3.25 sds sdsmapchars (sds *s*, const char * *from*, const char * *to*, size_t *setlen*)

Definition at line 1052 of file sds.c.

5.346.3.26 sds sdsnew (const char * *init*)

Definition at line 144 of file sds.c.

References sdsnewlen().

5.346.3.27 sds sdsnewlen (const void * *init*, size_t *initlen*)

Definition at line 81 of file sds.c.

References s_malloc, SDS_HDR_VAR, SDS_TYPE_16, SDS_TYPE_32, SDS_TYPE_5, SDS_TYPE_64, SDS_TYPE_8, and SDS_TYPE_BITS.

Referenced by dime_dntp_command_starttls(), libdime_base64_encode(), sdsdup(), sdsempty(), sdsfromlonglong(), sdsnew(), and sdssplitlen().

5.346.3.28 void sdsrange (sds *s*, int *start*, int *end*)

Definition at line 714 of file sds.c.

5.346.3.29 sds sdsRemoveFreeSpace (sds *s*)

Definition at line 245 of file sds.c.

References s_free, s_malloc, s_realloc, and SDS_TYPE_MASK.

5.346.3.30 sds* sdssplitargs (const char * *line*, int * *argc*)

Definition at line 933 of file sds.c.

References hex_digit_to_int(), is_hex_digit(), s_free, s_malloc, s_realloc, sdscatlen(), sdsempty(), and sdsfree().

5.346.3.31 sds* sdssplitlen (const char * *s*, int *len*, const char * *sep*, int *seplen*, int * *count*)

Definition at line 795 of file sds.c.

References `s_free`, `s_malloc`, `s_realloc`, `sdsfree()`, and `sdsnewlen()`.

5.346.3.32 void sdstolower (sds *s*)

Definition at line 743 of file sds.c.

5.346.3.33 void sdstoupper (sds *s*)

Definition at line 750 of file sds.c.

5.346.3.34 sds sdstrim (sds *s*, const char * *cset*)

Definition at line 683 of file sds.c.

5.346.3.35 void sdsupdatelen (sds *s*)

Definition at line 174 of file sds.c.

5.347 src/providers/dime/sds/sdsalloc.h File Reference

Defines

- #define [s_malloc](#) malloc
- #define [s_realloc](#) realloc
- #define [s_free](#) free

5.347.1 Define Documentation

5.347.1.1 #define s_free free

Definition at line 42 of file sdsalloc.h.

Referenced by sdscatvprintf(), sdsfree(), sdsfreesplitres(), sdsMakeRoomFor(), sdsRemoveFreeSpace(), sdssplitargs(), and sdssplitlen().

5.347.1.2 #define s_malloc malloc

Definition at line 40 of file sdsalloc.h.

Referenced by sdscatvprintf(), sdsMakeRoomFor(), sdsnewlen(), sdsRemoveFreeSpace(), sdssplitargs(), and sdssplitlen().

5.347.1.3 #define s_realloc realloc

Definition at line 41 of file sdsalloc.h.

Referenced by sdsMakeRoomFor(), sdsRemoveFreeSpace(), sdssplitargs(), and sdssplitlen().

5.348 src/providers/dime/sds/testhelp.h File Reference

Defines

- #define `test_cond`(descr, _c)
- #define `test_report`()

Variables

- int `__failed_tests` = 0
- int `__test_num` = 0

5.348.1 Define Documentation

5.348.1.1 #define test_cond(descr, _c)

Value:

```
do { \
    __test_num++; printf("%d - %s: ", __test_num, descr); \
    if(_c) printf("PASSED\n"); else {printf("FAILED\n"); __failed_tests++;} \
} while(0);
```

Definition at line 44 of file testhelp.h.

5.348.1.2 #define test_report()

Value:

```
do { \
    printf("%d tests, %d passed, %d failed\n", __test_num, \
        __test_num-__failed_tests, __failed_tests); \
    if (__failed_tests) { \
        printf("=== WARNING === We have failed tests here...\n"); \
        exit(1); \
    } \
} while(0);
```

Definition at line 48 of file testhelp.h.

5.348.2 Variable Documentation

5.348.2.1 int __failed_tests = 0

Definition at line 42 of file testhelp.h.

5.348.2.2 int __test_num = 0

Definition at line 43 of file testhelp.h.

5.349 src/providers/dime/signet-resolver/cache_pub.c File Reference

```
#include "dime/signet-resolver/cache.h"
```

Functions

- void * [get_cache_obj_data](#) (cached_object_t *object)
- void [destroy_cache_entry](#) (cached_object_t *entry)
- cached_object_t * [find_cached_object](#) (const char *oid, cached_store_t *store)
- cached_object_t * [find_cached_object_cmp](#) (const void *key, cached_store_t *store, cached_store_comparator_t cmpfn)
- int [cached_object_exists](#) (const unsigned char *hashid, cached_store_t *store)
- int [cached_object_exists_cmp](#) (const void *key, cached_store_t *store, cached_store_comparator_t cmpfn)
- cached_object_t * [add_cached_object](#) (const char *id, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)
- cached_object_t * [add_cached_object_forced](#) (const char *id, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)
- cached_object_t * [add_cached_object_cmp](#) (const char *id, const void *key, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, cached_store_comparator_t cmpfn)
- cached_object_t * [add_cached_object_cmp_forced](#) (const char *id, const void *key, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, cached_store_comparator_t cmpfn)
- int [remove_cached_object](#) (const char *oid, cached_store_t *store)
- int [remove_cached_object_cmp](#) (const void *key, cached_store_t *store, cached_store_comparator_t cmpfn)
- int [load_cache_contents](#) (void)
- int [save_cache_contents](#) (void)
- char * [get_dime_dir_location](#) (const char *suffix)
- char * [get_cache_location](#) (void)
- int [set_cache_location](#) (const char *path)
- int [set_cache_permissions](#) (unsigned long flags)

5.349.1 Function Documentation

5.349.1.1 cached_object_t* add_cached_object (const char *id, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed)

Definition at line 28 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.2 cached_object_t* add_cached_object_cmp (const char *id, const void *key, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, cached_store_comparator_t cmpfn)

Definition at line 36 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.3 cached_object_t* add_cached_object_cmp_forced (const char *id, const void *key, cached_store_t *store, unsigned long ttl, time_t expiration, void *data, int persists, int relaxed, cached_store_comparator_t cmpfn)

Definition at line 40 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.4 `cached_object_t* add_cached_object_forced (const char * id, cached_store_t * store, unsigned long tttl, time_t expiration, void * data, int persists, int relaxed)`

Definition at line 32 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.5 `int cached_object_exists (const unsigned char * hashid, cached_store_t * store)`

Definition at line 20 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.6 `int cached_object_exists_cmp (const void * key, cached_store_t * store, cached_store_comparator_t cmpfn)`

Definition at line 24 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.7 `void destroy_cache_entry (cached_object_t * entry)`

Definition at line 8 of file cache_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.349.1.8 `cached_object_t* find_cached_object (const char * oid, cached_store_t * store)`

Definition at line 12 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.9 `cached_object_t* find_cached_object_cmp (const void * key, cached_store_t * store, cached_store_comparator_t cmpfn)`

Definition at line 16 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.10 `char* get_cache_location (void)`

Definition at line 64 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.11 `void* get_cache_obj_data (cached_object_t * object)`

Definition at line 4 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.12 `char* get_dime_dir_location (const char * suffix)`

Definition at line 60 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.13 int load_cache_contents (void)

Definition at line 52 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.14 int remove_cached_object (const char * *oid*, cached_store_t * *store*)

Definition at line 44 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.15 int remove_cached_object_cmp (const void * *key*, cached_store_t * *store*, cached_store_comparator_t *cmpfn*)

Definition at line 48 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.16 int save_cache_contents (void)

Definition at line 56 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.17 int set_cache_location (const char * *path*)

Definition at line 68 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.349.1.18 int set_cache_permissions (unsigned long *flags*)

Definition at line 72 of file cache_pub.c.

References PUBLIC_FUNC_IMPL.

5.350 src/providers/dime/signet-resolver/dmtp.c File Reference

```
#include <openssl/x509v3.h>
#include "dime/signet-resolver/dmtp.h"
#include "dime/signet-resolver/cache.h"
#include "dime/signet-resolver/dns.h"
#include "dime/signet-resolver/mrec.h"
#include "dime/common/network.h"
#include "dime/common/misc.h"
#include "dime/common/error.h"
#include "providers/symbols.h"
```

Functions

- [signet_t * _get_signet](#) (const char *name, const char *fingerprint, int use_cache)
- [dmtp_session_t * _sgnt_resolv_dmtp_connect](#) (const char *domain, int force_family)
Establish a DMTP connection to the DX server of a provided dark domain.
- void [_sgnt_resolv_destroy_dmtp_session](#) (dmtp_session_t *session)
Destroy a DMTP session and its underlying data.
- [dmtp_session_t * _dx_connect_standard](#) (const char *host, const char *domain, int force_family, [dime_record_t](#) *dimerec)
Establish (force) a DMTP connection to a specified DX server on tcp port 26/ssl.
- [dmtp_session_t * _dx_connect_dual](#) (const char *host, const char *domain, int force_family, [dime_record_t](#) *dimerec, int failover)
Establish a DMTP connection to a specified DX server that is running DMTP in dual mode (port 25).
- int [_verify_dx_certificate](#) (dmtp_session_t *session)
Compare the TLS certificate presented by a remote host against the TLS certificate signature field of a DIME management record.
- char * [_sgnt_resolv_dmtp_get_signet](#) (dmtp_session_t *session, const char *signature, const char *fingerprint)
Look up a named user or organizational signet.
- int [_sgnt_resolv_dmtp_ehlo](#) (dmtp_session_t *session, const char *domain)
Issue an EHLO command to the remote DMTP server.
- int [_sgnt_resolv_dmtp_mail_from](#) (dmtp_session_t *session, const char *origin, size_t msgsize, [dmtp_mail_rettype_t](#) rettype, [dmtp_mail_datatype_t](#) dtype)
Issue a MAIL FROM command to the remote DMTP server.
- int [_sgnt_resolv_dmtp_rcpt_to](#) (dmtp_session_t *session, const char *domain)
Issue an RCPT TO command to the remote DMTP server.
- char * [_sgnt_resolv_dmtp_data](#) (dmtp_session_t *session, void *msg, size_t msglen)
Issue an DATA command to the remote DMTP server.
- int [_sgnt_resolv_dmtp_verify_signet](#) (dmtp_session_t *session, const char *signature, const char *fingerprint, char **newprint)
Verify whether a named user or org signet is current or not, given both its name and fingerprint.
- char * [_sgnt_resolv_dmtp_history](#) (dmtp_session_t *session, const char *signature, const char *startfp, const char *endfp)

Retrieve the history (key chain) of a named user or organizational signet.

- `char * _sgnt_resolv_dmtpl_stats (dmtpl_session_t *session, const unsigned char *secret __attribute__((__unused__)))`
- `dmtpl_mode_t _sgnt_resolv_dmtpl_str_to_mode (const char *modestr)`

Return the DMTP mode corresponding to a DMTP mode string.

- `dmtpl_mode_t _sgnt_resolv_dmtpl_get_mode (dmtpl_session_t *session)`

Get the current DMTP server mode.

- `int _sgnt_resolv_dmtpl_noop (dmtpl_session_t *session)`

Issue a no-operation command to the remote DMTP server.

- `int _sgnt_resolv_dmtpl_reset (dmtpl_session_t *session)`

Issue a reset command to the remote DMTP server.

- `char * _sgnt_resolv_dmtpl_help (dmtpl_session_t *session)`

Get a list of supported DMTP server commands by issuing the HELP command.

- `int _sgnt_resolv_dmtpl_quit (dmtpl_session_t *session, int do_close)`

Terminate an active DMTP session by issuing the QUIT command.

- `char * _sgnt_resolv_read_dmtpl_line (dmtpl_session_t *session, int *overflow, unsigned short *rcode, int *multiline)`

Read a CR/LF-terminated line of input from an active DMTP session.

- `char * _sgnt_resolv_read_dmtpl_multiline (dmtpl_session_t *session, int *overflow, unsigned short *rcode)`

Read a multiple-line response from the DMTP server.

- `char * _sgnt_resolv_parse_line_code (const char *line, unsigned short *rcode, int *multiline)`

Parse a DMTP line into a numerical code and the trailing text.

- `dmtpl_mode_t _sgnt_resolv_dmtpl_initiate_starttls (dmtpl_session_t *session, const char *dxname)`

Initiate a DMTP session over a plain TCP connection with the DMTP STARTTLS command.

- `int _sgnt_resolv_dmtpl_expect_banner (dmtpl_session_t *session)`

Attempt to receive a valid DMTP banner immediately after a connection is established.

- `int _sgnt_resolv_dmtpl_issue_command (dmtpl_session_t *session, const char *cmd)`

Issue a command to the remote server in a DMTP session.

- `char * _sgnt_resolv_dmtpl_send_and_read (dmtpl_session_t *session, const char *cmd, unsigned short *rcode)`

Issue a command to a DMTP server and get the response.

- `int _sgnt_resolv_dmtpl_write_data (dmtpl_session_t *session, const void *buf, size_t buflen)`

Write raw data to the remote end of a DMTP session.

5.350.1 Function Documentation

5.350.1.1 `dmtpl_session_t* _dx_connect_dual (const char *host, const char *domain, int force_family, dime_record_t *dimerec, int failover)`

Establish a DMTP connection to a specified DX server that is running DMTP in dual mode (port 25).

Note:

This function should only be called externally with care.

Parameters:

host the hostname of the DX server to which the DMTP connection will be established.

domain the dark domain which the DX server is servicing.

force_family an optional address family (AF_INET or AF_INET6) to force the TCP connection to take.

dimerec an optional pointer to a DIME management record to be attached to the session.

failover if set, attempt to retry failed connections on the backup port (587).

Returns:

NULL on failure, or a pointer to a newly established DMTP session on success.

Definition at line 338 of file dmtp.c.

References `_connect_host()`, `_dbgprint()`, `dmtp_session_t::_fd`, `_sgnt_resolv_destroy_dmtp_session()`, `_sgnt_resolv_dmtp_expect_banner()`, `_sgnt_resolv_dmtp_initiate_starttls()`, `dmtp_session_t::active`, `dmtp_mode_dmtp`, `dmtp_mode_dual`, `DMTP_PORT_DUAL`, `dmtp_session_t::domain`, `dmtp_session_t::drec`, `dmtp_session_t::dx`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dmtp_session_t::mode`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, and `RET_ERROR_PTR_FMT`.

Referenced by `_sgnt_resolv_dmtp_connect()`.

5.350.1.2 `dmtp_session_t* _dx_connect_standard (const char * host, const char * domain, int force_family, dime_record_t * dimerec)`

Establish (force) a DMTP connection to a specified DX server on tcp port 26/ssl.

Note:

This function should only be called externally with care.

Parameters:

host the hostname of the DX server to which the DMTP connection will be established.

domain the dark domain which the DX server is servicing.

force_family an optional address family (AF_INET or AF_INET6) to force the TCP connection to take.

dimerec an optional pointer to a DIME management record to be attached to the session.

Returns:

NULL on failure, or a pointer to a newly established DMTP session on success.

Definition at line 289 of file dmtp.c.

References `dmtp_session_t::_fd`, `_sgnt_resolv_destroy_dmtp_session()`, `_sgnt_resolv_dmtp_expect_banner()`, `_ssl_connect_host()`, `_ssl_disconnect()`, `dmtp_session_t::con`, `dmtp_mode_dmtp`, `DMTP_PORT`, `dmtp_session_t::domain`, `dmtp_session_t::drec`, `dmtp_session_t::dx`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dmtp_session_t::mode`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_sgnt_resolv_dmtp_connect()`.

5.350.1.3 `signet_t* _get_signet (const char * name, const char * fingerprint, int use_cache)`

Definition at line 15 of file dmtp.c.

References `_add_cached_object()`, `_clear_error_stack()`, `_dbgprint()`, `_find_cached_object()`, `_get_cache_obj_data()`, `_save_cache_contents()`, `_sgnt_resolv_destroy_dmtp_session()`, `_sgnt_resolv_dmtp_connect()`, `_sgnt_resolv_dmtp_get_signet()`, `_verify_dx_certificate()`, `cached_data_signet`, `cached_stores`, `dime_sgnt_signet_b64_deserialize()`, `dime_sgnt_signet_destroy()`, `dime_sgnt_validate_all()`, `dmtp_session_t::drec`, `dump_error_stack()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `get_last_error()`, `org_signet`, `dime_record_t::pubkey`, `RET_ERROR_PTR`, and `SS_FULL`.

5.350.1.4 void _sgnt_resolv_destroy_dmtplib_session (dmtplib_session_t * session)

Destroy a DMTP session and its underlying data.

Parameters:

session a pointer to the DMTP session to be destroyed/disconnected.

Definition at line 246 of file dmtplib.c.

References _destroy_dime_record(), dmtplib_session_t::fd, dmtplib_session_t::inbuf, _ssl_disconnect(), dmtplib_session_t::con, dmtplib_session_t::domain, dmtplib_session_t::drec, and dmtplib_session_t::dx.

Referenced by _dx_connect_dual(), _dx_connect_standard(), and _get_signet().

5.350.1.5 dmtplib_session_t* _sgnt_resolv_dmtplib_connect (const char * domain, int force_family)

Establish a DMTP connection to the DX server of a provided dark domain.

Note:

This function automatically queries the DIME management record of the domain to determine the appropriate way to establish a connection to the domain's DX server. This is the function that should be used by all general callers.

Parameters:

domain a null-terminated string containing the specified dark domain.

force_family an optional address family (AF_INET or AF_INET6) to force the TCP connection to take.

Returns:

NULL if unable to establish a DMTP connection successfully, or a pointer to the DMTP session on success.

Definition at line 158 of file dmtplib.c.

References _dbgprint(), _destroy_dime_record(), _dx_connect_dual(), _dx_connect_standard(), _get_dime_record(), _get_mx_records(), DMTP_MAX_MX_RETRIES, DMTP_PORT, DMTP_PORT_DUAL, dime_record_t::dx, ERR_BAD_PARAM, ERR_UNSPEC, RET_ERROR_PTR, and dime_record_t::validated.

Referenced by _get_signet().

5.350.1.6 char* _sgnt_resolv_dmtplib_data (dmtplib_session_t * session, void * msg, size_t msglen)

Issue an DATA command to the remote DMTP server.

Definition at line 772 of file dmtplib.c.

References _sgnt_resolv_dmtplib_send_and_read(), _sgnt_resolv_dmtplib_write_data(), _sgnt_resolv_read_dmtplib_line(), ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, PUSH_ERROR_FMT, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

5.350.1.7 int _sgnt_resolv_dmtplib_ahlo (dmtplib_session_t * session, const char * domain)

Issue an EHLO command to the remote DMTP server.

Parameters:

session a pointer to the DMTP session across which the EHLO command will be issued.

domain a null-terminated string containing the domain name that is the parameter to the EHLO command.

Returns:

0 if the EHLO command was issued and received successfully, or -1 on failure.

Definition at line 629 of file dmtp.c.

References `_sgnt_resolv_dmtp_issue_command()`, `_sgnt_resolv_read_dmtp_multiline()`, `_str_printf()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, and `RET_ERROR_INT`.

5.350.1.8 int _sgnt_resolv_dmtp_expect_banner (dmtp_session_t * session)

Attempt to receive a valid DMTP banner immediately after a connection is established.

Parameters:

session a pointer to the DMTP session to have the server banner read.

Returns:

-1 on failure or 0 if a banner was successfully received advertising DMTPv1 compatibility.

Definition at line 1667 of file dmtp.c.

References `_dbgprint()`, `_sgnt_resolv_read_dmtp_line()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, and `RET_ERROR_INT`.

Referenced by `_dx_connect_dual()`, and `_dx_connect_standard()`.

5.350.1.9 dmtp_mode_t _sgnt_resolv_dmtp_get_mode (dmtp_session_t * session)

Get the current DMTP server mode.

Parameters:

session a pointer to the DMTP session across which the MODE command will be issued.

Returns:

the value of the current DMTP mode on success, or `dmtp_mode_unknown` on failure.

Definition at line 1105 of file dmtp.c.

References `_sgnt_resolv_dmtp_send_and_read()`, `_sgnt_resolv_dmtp_str_to_mode()`, `dmtp_mode_unknown`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, and `RET_ERROR_CUST`.

5.350.1.10 char* _sgnt_resolv_dmtp_get_signet (dmtp_session_t * session, const char * signame, const char * fingerprint)

Look up a named user or organizational signet.

Note:

If the fingerprint parameter is omitted, the signet record returned by the server will be only for the current certificate.

Parameters:

session

signame the name of the requested organizational or user signet.

fingerprint if not NULL, an optional fingerprint that the returned signet MUST match.

Definition at line 551 of file dmtp.c.

References `_sgnt_resolv_dmtp_send_and_read()`, `chr_isspace`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_get_signet()`.

5.350.1.11 char* _sgnt_resolv_dmtplib_help (dmtplib_session_t * session)

Get a list of supported DMTP server commands by issuing the HELP command.

Parameters:

session a pointer to the DMTP session across which the HELP command will be issued.

Returns:

NULL on failure or a pointer to a string containing the full server help table on success.

Definition at line 1207 of file dmtplib.c.

References _sgnt_resolv_dmtplib_issue_command(), _sgnt_resolv_read_dmtplib_multiline(), ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_FMT, and RET_ERROR_PTR.

5.350.1.12 char* _sgnt_resolv_dmtplib_history (dmtplib_session_t * session, const char * signame, const char * startfp, const char * endfp)

Retrieve the history (key chain) of a named user or organizational signet.

Parameters:

session a pointer to the DMTP session across which the HIST command will be issued.

signame the name of the user or organizational signet to have its history retrieved.

startfp the starting fingerprint of the signet in the queried chain of custody.

endfp the ending fingerprint of the signet in the queried chain of custody.

Returns:

NULL on failure, or a pointer to a string containing the signet's history on success.

Definition at line 941 of file dmtplib.c.

References _sgnt_resolv_dmtplib_issue_command(), _sgnt_resolv_read_dmtplib_multiline(), ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_FMT, PUSH_ERROR_SYSCALL, and RET_ERROR_PTR.

5.350.1.13 dmtplib_mode_t _sgnt_resolv_dmtplib_initiate_starttls (dmtplib_session_t * session, const char * dxname)

Initiate a DMTP session over a plain TCP connection with the DMTP STARTTLS command.

Note:

The dxname argument will be used as a parameter to the STARTTLS command for requesting a TLS certificate explicitly by hostname.

Parameters:

session a pointer to the DMTP session to have its transport security upgraded with TLS.

dxname the name of the DX server providing the TLS service of this DMTP session.

Returns:

dmtplib_mode_unknown on failure or the active mode of the DMTP session as reported by the server on success.

Definition at line 1585 of file dmtplib.c.

References dmtplib_session_t::fd, _sgnt_resolv_dmtplib_str_to_mode(), _sgnt_resolv_read_dmtplib_line(), _sgnt_resolv_read_dmtplib_multiline(), _ssl_starttls(), chr_isspace, dmtplib_session_t::con, dmtplib_mode_dmtplib, dmtplib_mode_unknown, ERR_BAD_PARAM, ERR_UNSPEC, dmtplib_session_t::mode, PUSH_ERROR_FMT, and RET_ERROR_CUST.

Referenced by _dx_connect_dual().

5.350.1.14 int _sgnt_resolv_dmtp_issue_command (dmtp_session_t * session, const char * cmd)

Issue a command to the remote server in a DMTP session.

Parameters:

session a pointer to the DMTP session across which the command will be issued.

cmd a null-terminated string containing the value of the specified DMTP command.

Returns:

-1 on failure or 0 if the command was successfully sent.

Definition at line 1715 of file dmtp.c.

References `_dbgprint()`, `dmtp_session_t::fd`, `dmtp_session_t::con`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, and `SSL_write_d`.

Referenced by `_sgnt_resolv_dmtp_ehlo()`, `_sgnt_resolv_dmtp_help()`, `_sgnt_resolv_dmtp_history()`, `_sgnt_resolv_dmtp_send_and_read()`, and `_sgnt_resolv_dmtp_stats()`.

5.350.1.15 int _sgnt_resolv_dmtp_mail_from (dmtp_session_t * session, const char * origin, size_t msgsize, dmtp_mail_rettype_t rettype, dmtp_mail_datatype_t dtype)

Issue a MAIL FROM command to the remote DMTP server.

Definition at line 667 of file dmtp.c.

References `_sgnt_resolv_dmtp_send_and_read()`, `_str_printf()`, `data_type_7bit`, `data_type_8bit`, `data_type_default`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, `RET_ERROR_INT`, `return_type_default`, `return_type_display`, `return_type_full`, and `return_type_header`.

5.350.1.16 int _sgnt_resolv_dmtp_noop (dmtp_session_t * session)

Issue a no-operation command to the remote DMTP server.

Parameters:

session a pointer to the DMTP session across which the NOOP command will be issued.

Returns:

-1 on failure or 0 if the command was successfully executed.

Definition at line 1151 of file dmtp.c.

References `_sgnt_resolv_dmtp_send_and_read()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, and `RET_ERROR_INT`.

5.350.1.17 int _sgnt_resolv_dmtp_quit (dmtp_session_t * session, int do_close)

Terminate an active DMTP session by issuing the QUIT command.

Parameters:

session a pointer to the DMTP session across which the QUIT command will be issued.

do_close currently unused

Returns:

-1 on failure or 0 if the command was successfully executed.

Definition at line 1238 of file dmtplib.c.

References `dmtplib_session_t::fd`, `_sgnt_resolv_dmtplib_send_and_read()`, `_ssl_disconnect()`, `dmtplib_session_t::active`, `dmtplib_session_t::con`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR`, `PUSH_ERROR_FMT`, and `RET_ERROR_INT`.

5.350.1.18 `int _sgnt_resolv_dmtplib_rcpt_to (dmtplib_session_t * session, const char * domain)`

Issue an RCPT TO command to the remote DMTP server.

Parameters:

session a pointer to the DMTP session across which the RCPT TO command will be issued.

domain a null-terminated string containing the domain name that is the parameter to the RCPT TO command.

Returns:

0 if the RCPT TO command was issued and received successfully, or -1 on failure.

Definition at line 739 of file dmtplib.c.

References `_sgnt_resolv_dmtplib_send_and_read()`, `_str_printf()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, and `RET_ERROR_INT`.

5.350.1.19 `int _sgnt_resolv_dmtplib_reset (dmtplib_session_t * session)`

Issue a reset command to the remote DMTP server.

Parameters:

session a pointer to the DMTP session across which the RSET command will be issued.

Returns:

-1 on failure or 0 if the command was successfully executed.

Definition at line 1179 of file dmtplib.c.

References `_sgnt_resolv_dmtplib_send_and_read()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, and `RET_ERROR_INT`.

5.350.1.20 `char* _sgnt_resolv_dmtplib_send_and_read (dmtplib_session_t * session, const char * cmd, unsigned short * rcode)`

Issue a command to a DMTP server and get the response.

Parameters:

session a pointer to the DMTP session across which the command will be issued.

cmd a null-terminated string containing the value of the specified DMTP command.

rcode

Returns:

NULL on failure or a pointer to a string containing the next line(s) of output from the DMTP server on success.

Definition at line 1756 of file dmtplib.c.

References `_sgnt_resolv_dmtplib_issue_command()`, `_sgnt_resolv_read_dmtplib_line()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, and `RET_ERROR_PTR`.

Referenced by `_sgnt_resolv_dmtplib_data()`, `_sgnt_resolv_dmtplib_get_mode()`, `_sgnt_resolv_dmtplib_get_signet()`, `_sgnt_resolv_dmtplib_mail_from()`, `_sgnt_resolv_dmtplib_noop()`, `_sgnt_resolv_dmtplib_quit()`, `_sgnt_resolv_dmtplib_rcpt_to()`, `_sgnt_resolv_dmtplib_reset()`, and `_sgnt_resolv_dmtplib_verify_signet()`.

5.350.1.21 `char* _sgnt_resolv_dmtp_stats (dmtp_session_t * session, const unsigned char *secret __attribute__((__unused__)))`

Definition at line 991 of file dmtp.c.

References `_sgnt_resolv_dmtp_issue_command()`, `_sgnt_resolv_read_dmtp_multiline()`, `_str_printf()`, `chr_isspace`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, and `RET_ERROR_PTR`.

5.350.1.22 `dmtp_mode_t _sgnt_resolv_dmtp_str_to_mode (const char * modestr)`

Return the DMTP mode corresponding to a DMTP mode string.

Parameters:

modestr a null-terminated string containing the human-readable name of the mode.

Returns:

the DMTP mode type corresponding to the supplied mode string, or `dmtp_mode_unknown` otherwise.

Definition at line 1083 of file dmtp.c.

References `dmtp_mode_dmtp`, `dmtp_mode_esmtp`, `dmtp_mode_smtp`, and `dmtp_mode_unknown`.

Referenced by `_sgnt_resolv_dmtp_get_mode()`, and `_sgnt_resolv_dmtp_initiate_starttls()`.

5.350.1.23 `int _sgnt_resolv_dmtp_verify_signet (dmtp_session_t * session, const char * signame, const char * fingerprint, char ** newprint)`

Verify whether a named user or org signet is current or not, given both its name and fingerprint.

Parameters:

session a pointer to the DMTP session across which the VRFY command will be issued.

signame the name of the user or organizational signet to be verified.

fingerprint the fingerprint of the signet to be checked for updates.

newprint an optional pointer to a string which will be updated to point to the latest fingerprint of the specified signet, if it is out of date.

Returns:

-1 on general failure, 0 if the signet fingerprint was out of date, or 1 if it is the most current one.

Definition at line 870 of file dmtp.c.

References `_sgnt_resolv_dmtp_send_and_read()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, and `status`.

5.350.1.24 `int _sgnt_resolv_dmtp_write_data (dmtp_session_t * session, const void * buf, size_t buflen)`

Write raw data to the remote end of a DMTP session.

Parameters:

session a pointer to the DMTP session to which the data will be written.

buf a pointer to the buffer containing the raw data to be written.

buflen the size, in bytes, of the data buffer to be written to the DMTP session.

Returns:

0 if all requested bytes were written successfully to the connection, or -1 on failure.

Definition at line 1783 of file dmtplib.c.

References `_dbgprint()`, `dmtplib_session_t::fd`, `dmtplib_session_t::con`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, and `SSL_write_d`.

Referenced by `_sgnt_resolv_dmtplib_data()`.

5.350.1.25 `char* _sgnt_resolv_parse_line_code (const char * line, unsigned short * rcode, int * multiline)`

Parse a DMTP line into a numerical code and the trailing text.

Parameters:

line a pointer to a null-terminated string to be parsed as a DMTP response line or lines.

rcode an opointer to a variable that will receive the numeric response code of the DMTP reply.

multiline an optional parameter that if set will permit multiline DMTP responses. If this was the final line of a multiline response, the value will be set to 1 when the function returns, or 0 if there is more content to follow.

Returns:

NULL on failure or a pointer to a string containing the next line(s) of output from the DMTP server.

Definition at line 1524 of file dmtplib.c.

References `chr_isspace`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `RET_ERROR_PTR`, and `RET_ERROR_PTR_FMT`.

Referenced by `_sgnt_resolv_read_dmtplib_line()`.

5.350.1.26 `char* _sgnt_resolv_read_dmtplib_line (dmtplib_session_t * session, int * overflow, unsigned short * rcode, int * multiline)`

Read a CR/LF-terminated line of input from an active DMTP session.

Parameters:

session a pointer to the DMTP session from which the response line will be read.

overflow an optional pointer to a variable that will be set if the read operation exceeds the size of the internal line buffer.

rcode an optional pointer to a variable that will receive the numeric response code of the DMTP reply that was just received.

multiline an optional parameter that if set will permit multiline DMTP responses. If this was the final line of a multiline response, the value will be set to 1 when the function returns, or 0 if there is more content to follow.

Returns:

NULL on failure or a pointer to a string containing the next line(s) of output from the DMTP server.

Definition at line 1292 of file dmtplib.c.

References `_dbgprint()`, `dmtplib_session_t::fd`, `dmtplib_session_t::inbuf`, `dmtplib_session_t::inpos`, `_sgnt_resolv_parse_line_code()`, `_ssl_disconnect()`, `dmtplib_session_t::active`, `dmtplib_session_t::con`, `dmtplib_mode_dmtplib`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dmtplib_session_t::mode`, `PUSH_ERROR_OPENSSL`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, and `SSL_read_d`.

Referenced by `_sgnt_resolv_dmtplib_data()`, `_sgnt_resolv_dmtplib_expect_banner()`, `_sgnt_resolv_dmtplib_initiate_starttls()`, `_sgnt_resolv_dmtplib_send_and_read()`, and `_sgnt_resolv_read_dmtplib_multiline()`.

5.350.1.27 `char* _sgnt_resolv_read_dmtplib_multiline (dmtplib_session_t * session, int * overflow, unsigned short * rcode)`

Read a multiple-line response from the DMTP server.

Note:

A multiline response is designated by the presence of a hyphen between the response code and the response text. A regular response, or the final line of a multiline response will instead have a space.

Parameters:

session a pointer to the DMTP session from which the response line(s) will be read.

overflow an optional pointer to a variable that will be set if the read operation exceeds the size of the internal line buffer.

rcode an optional pointer to a variable that will receive the numeric response code of the DMTP reply that was just received.

Returns:

NULL on failure or a pointer to a string containing the next response from the DMTP server on success.

Definition at line 1443 of file dmtp.c.

References `_sgnt_resolv_read_dmtp_line()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_sgnt_resolv_dmtp_ehlo()`, `_sgnt_resolv_dmtp_help()`, `_sgnt_resolv_dmtp_history()`, `_sgnt_resolv_dmtp_initiate_starttls()`, and `_sgnt_resolv_dmtp_stats()`.

5.350.1.28 int _verify_dx_certificate (dmtp_session_t * session)

Compare the TLS certificate presented by a remote host against the TLS certificate signature field of a DIME management record.

Note:

This function accomplishes a number of steps in the following order: 1. If the DIME record has a TLS certificate signature field, perform a comparison of it against the remote peer's certificate. 2. If necessary, perform x509 chain validation against the certificate. 3. If necessary, verify that the certificate has not been revoked by issuing an OCSP request.

Parameters:

session a pointer to the DMTP session to have its TLS certificate verified.

Returns:

-1 on general error, 0 if the connection's TLS certificate failed verification, and 1 if the certificate passed verification.

Definition at line 404 of file dmtp.c.

References `_dbgprint()`, `_do_ocsp_validation()`, `_do_x509_hostname_check()`, `_do_x509_validation()`, `_ed25519_verify_sig()`, `_get_x509_cert_sha_hash()`, `dmtp_session_t::con`, `dmtp_session_t::drec`, `dmtp_session_t::dx`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `dime_record_t::pubkey`, `ED25519_KEY::public_key`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_INT`, `SHA_512_SIZE`, `SSL_get_peer_cert_chain_d`, `SSL_get_peer_certificate_d`, `dime_record_t::tlssig`, `dime_record_t::validated`, and `X509_check_issued_d`.

Referenced by `_get_signet()`.

5.351 src/servers/dmtp/dmtp.c File Reference

```
#include "magma.h"
```

Functions

- void [dmtp_ehlo](#) ([connection_t](#) *con)
Process an DMTP EHLO command.
- void [dmtp_helo](#) ([connection_t](#) *con)
Process an DMTP HELO command.
- void [dmtp_noop](#) ([connection_t](#) *con)
Perform an DMTP NOOP (no-operation) command.
- void [dmtp_invalid](#) ([connection_t](#) *con)
A function that is executed when an invalid DMTP command is executed.
- void [dmtp_quit](#) ([connection_t](#) *con)
Gracefully terminate an DMTP session, especially in response to an DMTP QUIT command.
- void [dmtp_rset](#) ([connection_t](#) *con)
Reset the DMTP session, in response to an DMTP RSET command.
- void [dmtp_rcpt](#) ([connection_t](#) *con)
Specify the destination domain for a message in response to an DMTP RCPT command.
- void [dmtp_mail](#) ([connection_t](#) *con)
Specify the origin domain for a message in response to an DMTP MAIL command.
- void [dmtp_data](#) ([connection_t](#) *con)
Process an DMTP MAIL command.
- void [dmtp_mode](#) ([connection_t](#) *con)
- void [dmtp_sgnt](#) ([connection_t](#) *con)
- void [dmtp_hist](#) ([connection_t](#) *con)
- void [dmtp_vrfy](#) ([connection_t](#) *con)
- void [dmtp_help](#) ([connection_t](#) *con)
- void [dmtp_verb](#) ([connection_t](#) *con)
- void [dmtp_init](#) ([connection_t](#) *con)
The start of the protocol handler for the DMTP server.

5.351.1 Function Documentation

5.351.1.1 void dmtp_data (connection_t * con)

Process an DMTP MAIL command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 123 of file dmtp.c.

References `con_write_bl()`, and `dmtp_requeue()`.

Referenced by `dmtp_process()`.

5.351.1.2 void dmtp_ehlo (connection_t * con)

Process an DMTP EHLO command. dmtp.c

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 15 of file dmtp.c.

References `con_print()`, `st_char_get()`, and `st_length_int()`.

5.351.1.3 void dmtp_helo (connection_t * con)

Process an DMTP HELO command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 28 of file dmtp.c.

References `con_print()`, `st_char_get()`, and `st_length_int()`.

5.351.1.4 void dmtp_help (connection_t * con)

Definition at line 158 of file dmtp.c.

References `con_write_bl()`.

5.351.1.5 void dmtp_hist (connection_t * con)

Definition at line 144 of file dmtp.c.

References `con_write_bl()`.

5.351.1.6 void dmtp_init (connection_t * con)

The start of the protocol handler for the DMTP server.

Parameters:

con the new inbound DMTP client connection.

Returns:

This function returns no value.

Definition at line 175 of file dmtplib.c.

References `con_print()`, `con_reverse_enqueue()`, `dmtplib_requeue()`, `st_char_get()`, and `st_length_int()`.

Referenced by `protocol_enqueue()`.

5.351.1.7 void dmtplib_invalid (connection_t * con)

A function that is executed when an invalid DMTP command is executed.

Returns:

This function returns no value.

Definition at line 52 of file dmtplib.c.

References `con_write_bl()`.

Referenced by `dmtplib_process()`.

5.351.1.8 void dmtplib_mail (connection_t * con)

Specify the origin domain for a message in response to an DMTP MAIL command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 111 of file dmtplib.c.

References `con_write_bl()`.

5.351.1.9 void dmtplib_mode (connection_t * con)

Definition at line 130 of file dmtplib.c.

References `con_write_bl()`.

5.351.1.10 void dmtplib_noop (connection_t * con)

Perform an DMTP NOOP (no-operation) command.

Note:

This command does essentially nothing and is mostly a way to keep connections alive without timing out due to inactivity.

Returns:

This function returns no value.

Definition at line 42 of file dmtp.c.

References `con_write_bl()`.

5.351.1.11 void dmtp_quit (connection_t * con)

Gracefully terminate an DMTP session, especially in response to an DMTP QUIT command.

Parameters:

the DMTP client connection to be terminated.

Returns:

This function returns no value.

Definition at line 65 of file dmtp.c.

References `con_destroy()`, `con_status()`, and `con_write_bl()`.

Referenced by `dmtp_process()`, and `dmtp_requeue()`.

5.351.1.12 void dmtp_rcpt (connection_t * con)

Specify the destination domain for a message in response to an DMTP RCPT command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 100 of file dmtp.c.

References `con_write_bl()`.

5.351.1.13 void dmtp_rset (connection_t * con)

Reset the DMTP session, in response to an DMTP RSET command.

Note:

This command clears any sender, recipient, and mail data, along with all buffers and state tables. The connection structure is reset to the same it was in immediately after the HELO/EHLO command.

Parameters:

con the DMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 88 of file dmtp.c.

References `con_write_bl()`, and `dmtp_session_reset()`.

5.351.1.14 void dmtplib_sgt (connection_t * con)

Definition at line 137 of file dmtplib.c.

References `con_write_bl()`.

5.351.1.15 void dmtplib_verb (connection_t * con)

Definition at line 164 of file dmtplib.c.

References `con_write_bl()`.

5.351.1.16 void dmtplib_vrfy (connection_t * con)

Definition at line 151 of file dmtplib.c.

References `con_write_bl()`.

5.352 src/providers/dime/signet-resolver/dmtp_pub.c File Reference

```
#include "dime/signet-resolver/dmtp.h"
```

Functions

- [signet_t * get_signet](#) (const char *name, const char *fingerprint, int use_cache)
- [dmtp_session_t * sgnt_resolv_dmtp_connect](#) (const char *domain, int force_family)
- void [sgnt_resolv_destroy_dmtp_session](#) (dmtp_session_t *session)
- [dmtp_session_t * dx_connect_standard](#) (const char *host, const char *domain, int force_family, [dime_record_t](#) *dimerec)
- [dmtp_session_t * dx_connect_dual](#) (const char *host, const char *domain, int force_family, [dime_record_t](#) *dimerec, int failover)
- int [verify_dx_certificate](#) (dmtp_session_t *session)
- char * [sgnt_resolv_dmtp_get_signet](#) (dmtp_session_t *session, const char *signature, const char *fingerprint)
- int [sgnt_resolv_dmtp_ehlo](#) (dmtp_session_t *session, const char *domain)
- int [sgnt_resolv_dmtp_mail_from](#) (dmtp_session_t *session, const char *origin, size_t msgsize, [dmtp_mail_rettype_t](#) rettype, [dmtp_mail_datatype_t](#) dtype)
- int [sgnt_resolv_dmtp_rcpt_to](#) (dmtp_session_t *session, const char *domain)
- char * [sgnt_resolv_dmtp_data](#) (dmtp_session_t *session, void *msg, size_t msglen)
- int [sgnt_resolv_dmtp_verify_signet](#) (dmtp_session_t *session, const char *signature, const char *fingerprint, char **newprint)
- char * [sgnt_resolv_dmtp_history](#) (dmtp_session_t *session, const char *signature, const char *startfp, const char *endfp)
- char * [sgnt_resolv_dmtp_stats](#) (dmtp_session_t *session, const unsigned char *secret)
- [dmtp_mode_t](#) [sgnt_resolv_dmtp_str_to_mode](#) (const char *modestr)
- [dmtp_mode_t](#) [sgnt_resolv_dmtp_get_mode](#) (dmtp_session_t *session)
- int [sgnt_resolv_dmtp_noop](#) (dmtp_session_t *session)
- int [sgnt_resolv_dmtp_reset](#) (dmtp_session_t *session)
- char * [sgnt_resolv_dmtp_help](#) (dmtp_session_t *session)
- int [sgnt_resolv_dmtp_quit](#) (dmtp_session_t *session, int do_close)

5.352.1 Function Documentation

5.352.1.1 dmtp_session_t* dx_connect_dual (const char * host, const char * domain, int force_family, dime_record_t * dimerec, int failover)

Definition at line 20 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.2 dmtp_session_t* dx_connect_standard (const char * host, const char * domain, int force_family, dime_record_t * dimerec)

Definition at line 16 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.3 signet_t* get_signet (const char * name, const char * fingerprint, int use_cache)

Definition at line 4 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.4 void sgnt_resolv_destroy_dmtp_session (dmtp_session_t * session)

Definition at line 12 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.352.1.5 dmtplib_session_t* sgnt_resolv_dmtplib_connect (const char * *domain*, int *force_family*)

Definition at line 8 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.6 char* sgnt_resolv_dmtplib_data (dmtplib_session_t * *session*, void * *msg*, size_t *msglen*)

Definition at line 44 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.7 int sgnt_resolv_dmtplib_ehlo (dmtplib_session_t * *session*, const char * *domain*)

Definition at line 32 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.8 dmtplib_mode_t sgnt_resolv_dmtplib_get_mode (dmtplib_session_t * *session*)

Definition at line 64 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.9 char* sgnt_resolv_dmtplib_get_signet (dmtplib_session_t * *session*, const char * *signature*, const char * *fingerprint*)

Definition at line 28 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.10 char* sgnt_resolv_dmtplib_help (dmtplib_session_t * *session*)

Definition at line 76 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.11 char* sgnt_resolv_dmtplib_history (dmtplib_session_t * *session*, const char * *signature*, const char * *startfp*, const char * *endfp*)

Definition at line 52 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.12 int sgnt_resolv_dmtplib_mail_from (dmtplib_session_t * *session*, const char * *origin*, size_t *msgsize*, dmtplib_mail_rettype_t *rettype*, dmtplib_mail_datatype_t *dtype*)

Definition at line 36 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.13 int sgnt_resolv_dmtplib_noop (dmtplib_session_t * *session*)

Definition at line 68 of file dmtplib_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.14 int sgnt_resolv_dmtp_quit (dmtp_session_t * session, int do_close)

Definition at line 80 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.15 int sgnt_resolv_dmtp_rcpt_to (dmtp_session_t * session, const char * domain)

Definition at line 40 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.16 int sgnt_resolv_dmtp_reset (dmtp_session_t * session)

Definition at line 72 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.17 char* sgnt_resolv_dmtp_stats (dmtp_session_t * session, const unsigned char * secret)

Definition at line 56 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.18 dmtp_mode_t sgnt_resolv_dmtp_str_to_mode (const char * modestr)

Definition at line 60 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.19 int sgnt_resolv_dmtp_verify_signet (dmtp_session_t * session, const char * signame, const char * fingerprint, char ** newprint)

Definition at line 48 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.352.1.20 int verify_dx_certificate (dmtp_session_t * session)

Definition at line 24 of file dmtp_pub.c.

References PUBLIC_FUNC_IMPL.

5.353 src/providers/dime/signet-resolver/dns.c File Reference

```
#include "dime/signet-resolver/dns.h"
#include "dime/signet-resolver/cache.h"
#include "dime/common/misc.h"
#include "dime/common/error.h"
#include "providers/symbols.h"
```

Defines

- #define [INITIALIZE_DNS](#)() { if (!_dns_initialized && (_initialize_resolver() < 0)) { RET_ERROR_PTR(ERR_UNSPEC, "failed to initialize DNS resolver"); } }

Functions

- [size_t _mem_append_canon](#) (unsigned char **buf, size_t *blen, const char *name)
Append a DNS label in uncompressed, canonical format to a dynamic buffer.
- unsigned int [_get_keytag](#) (const unsigned char *rdata, size_t rdlen)
Calculate the keytag value for a DNSKEY resource record.
- RSA * [_get_rsa_dnskey](#) (const unsigned char *rdptr, size_t rdlen)
Extract an RSA public key from a DNSKEY resource record.
- int [_rsa_verify_record](#) (const char *label, unsigned char algorithm, RSA *pubkey, const unsigned char *rrsig, const unsigned char *sigbuf, size_t siglen, ns_msg *dhandle)
Verify that an RRSIG record has been signed properly using a given RSA public key.
- int [_load_dnskey_file](#) (const char *filename)
Load a collection of DNS key(s) from a config file (into the object cache).
- int [_is_validated_key](#) (dnskey_t *dk)
Trace a DNSKEY backwards through its DS entry to the root, to see if it is validated.
- int [_compute_dnskey_sha_hash](#) (const dnskey_t *key, size_t nbits, unsigned char *outbuf)
Perform a SHA-family hash against a DNSKEY public key.
- dnskey_t * [_add_dnskey_entry_rsa](#) (const char *label, uint16_t flags, unsigned char algorithm, RSA *pubkey, unsigned int keytag, const unsigned char *rdata, size_t rdlen, unsigned long ttl, unsigned int do_cache, int forced)
Create and add a new RSA DNSKEY entry to the object cache.
- ds_t * [_add_ds_entry](#) (const char *label, unsigned int keytag, unsigned char algorithm, unsigned char digest_type, const unsigned char *digest, size_t dlen, unsigned long ttl)
Create and add a new DS entry to the object cache.
- dnskey_t * [_get_dnskey_by_tag](#) (unsigned int tag, const char *signer, int force_lookup)
Find a DNSKEY entry by its keytag value.
- ds_t * [_get_ds_by_dnskey](#) (const dnskey_t *key)
Look up a DS record by its matching DNSKEY record.

- `int _dnskey_tag_comparator` (const void *object, const void *key)
An object cache callback for comparing two DNSKEY records based on tag value.
- `int _dnskey_domain_comparator` (const void *object, const void *key)
An object cache callback for comparing two DNSKEY records based on domain label.
- `int _ds_comparator` (const void *object, const void *key)
An object cache callback for comparing two DS records for general equivalence.
- `void _destroy_dnskey` (dnskey_t *key)
Free a DNSKEY object and its underlying data.
- `void _destroy_ds` (ds_t *ds)
Free a DS object and its underlying data.
- `dnskey_t * _add_dnskey_entry` (const char *label, const unsigned char *buf, size_t len, unsigned long ttl)
Create and add a new DNSKEY entry to the object cache.
- `int _validate_rrsig_rr` (const char *label, ns_msg *dhandle, unsigned short covered, const unsigned char *rdata, size_t rdlen, dnskey_t **outkey)
Verify the signature provided by an RRSIG record.
- `void _dump_dns_header` (ns_msg *handle)
Dump the contents of a DNS reply header.
- `void * _lookup_dnskey` (const char *label)
- `void * _lookup_ds` (const char *label)
Lookup the DS record for a specified domain.
- `char * _get_txt_record` (const char *qstring, unsigned long *ttl, int *validated)
Get the answer to a DNS TXT record query.
- `void _free_mx_records` (mx_record_t **mxs)
Free a collection of MX records returned by `_get_mx_records()`.
- `mx_record_t ** _get_mx_records` (const char *qstring)
Retrieve the collection of MX records for a given domain.
- `int _initialize_resolver` (void)
Initialize the DNS resolver subsystem.
- `void _destroy_dnskey_record_cb` (void *record)
A callback handler to destroy a DNSKEY record.
- `void _dump_dnskey_record_cb` (FILE *fp, void *record, int brief)
A callback handler to dump a DNSKEY record to the console.
- `void _destroy_ds_record_cb` (void *record)
A callback handler to destroy a DS record.
- `void _dump_ds_record_cb` (FILE *fp, void *record, int brief)
A callback handler to dump a DS record to a file.

- void * [_deserialize_ds_record_cb](#) (void *data, size_t len)
A callback handler to deserialize a DS record from the object cache.
- void * [_serialize_ds_record_cb](#) (void *record, size_t *outlen)
A callback record to serialize a DS record for storage in the object cache.
- void * [_deserialize_dnskey_record_cb](#) (void *data, size_t len)
A callback handler to deserialize a DNSKEY record.
- void * [_serialize_dnskey_record_cb](#) (void *record, size_t *outlen)
A callback handler to serialize a DNSKEY record.
- void * [_clone_dnskey_record_cb](#) (void *record)
- void [_fixup_ds_links](#) (void)
Link any DNSKEY entries in the cache with dangling DS pointers to the right place.
- void [_fixup_dnskey_validation](#) (void)
Mark as validated any DNSKEY entries in the object cache that should be but aren't.
- int [_sort_rrs_canonical](#) (ns_msg *dhandle, int *ordbuf, size_t nanswers)
Sort the RRs in an RRSET into canonical order for DNSSEC signature verification.
- int [_compare_rdata](#) (const unsigned char *rdata1, size_t rlen1, const unsigned char *rdata2, size_t rlen2)
Compare two RR's, for the purpose of allowing for canonical ordering within an RRSET.
- char * [_get_signing_key_names](#) (dnskey_t **keys)
Get the names of a collection of signing DNSKEYs for human display.

5.353.1 Define Documentation

- 5.353.1.1** `#define INITIALIZE_DNS() { if (!_dns_initialized && (_initialize_resolver() < 0)) { RET_ERROR_PTR(ERR_UNSPEC, "failed to initialize DNS resolver"); } }`

Definition at line 9 of file dns.c.

Referenced by `_get_mx_records()`, and `_get_txt_record()`.

5.353.2 Function Documentation

- 5.353.2.1** `dnskey_t* _add_dnskey_entry (const char * label, const unsigned char * buf, size_t len, unsigned long ttl)`

Create and add a new DNSKEY entry to the object cache.

Parameters:

label a pointer to a null-terminated string containing the name of the domain supplying the specified key.

buf a pointer to the beginning of the DNSKEY's RR rdata.

len the length, in bytes, of the DNSKEY RR's rdata.

ttl the TTL value indicated by the DNSKEY resource record.

Definition at line 1140 of file dns.c.

References `_add_dnskey_entry_rsa()`, `_dbgprint()`, `_fixup_dnskey_validation()`, `_get_keytag()`, `_get_rsa_dnskey()`, `dnskey_rr_flags_t`, `DNSKEY_RR_PROTO`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `NS_ALG_RSASHA1`, `NS_ALG_RSASHA256`, `NS_ALG_RSASHA512`, `RET_ERROR_PTR`, and `RET_ERROR_PTR_FMT`.

Referenced by `_lookup_dnskey()`.

5.353.2.2 `dnskey_t* _add_dnskey_entry_rsa (const char *label, uint16_t flags, unsigned char algorithm, RSA *pubkey, unsigned int keytag, const unsigned char *rdata, size_t rdlen, unsigned long ttl, unsigned int do_cache, int forced)`

Create and add a new RSA DNSKEY entry to the object cache.

Parameters:

label a pointer to a null-terminated string containing the name of the domain supplying the specified key.

flags a bitmask of flags corresponding to the DNSKEY's RR flags field.

algorithm the value of the DNSKEY's RR algorithm field.

pubkey a pointer to an RSA public key used to validate signatures made by this key.

keytag a numerical value linking this to other RRSIG or DS records.

rdata a pointer to the beginning of this DNSKEY's RR rdata.

rdlen the length, in bytes, of the DNSKEY's RR rdata.

ttl the TTL value indicated by the DNSKEY resource record.

do_cache if set, the new object cache entry for this DNSKEY can be persisted to the cache file.

forced if set, this entry will forcibly replace (overshadow) any clashing or matching existing DNSKEY entry inside the object cache.

Returns:

NULL on failure, or a pointer to the new DNSKEY object cache entry on success.

Definition at line 698 of file dns.c.

References `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_destroy_dnskey()`, `_dnskey_tag_comparator()`, `_get_cache_obj_data()`, `dnskey::algorithm`, `cached_data_dnskey`, `cached_stores`, `DNSKEY_RR_FLAG_SEP`, `DNSKEY_RR_FLAG_ZK`, `dnskey::do_cache`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `dnskey::pubkey`, `PUSH_ERROR_SYSCALL`, `dnskey::rdata`, `dnskey::rdlen`, and `RET_ERROR_PTR`.

Referenced by `_add_dnskey_entry()`, and `_load_dnskey_file()`.

5.353.2.3 `ds_t* _add_ds_entry (const char *label, unsigned int keytag, unsigned char algorithm, unsigned char digest_type, const unsigned char *digest, size_t dlen, unsigned long ttl)`

Create and add a new DS entry to the object cache.

Parameters:

label a pointer to a null-terminated string containing the name of the domain supplying the specified DS record.

keytag a numerical value identifying the DNSKEY entry linked to this record.

algorithm the value of the DS's RR algorithm field.

digest_type the value of the DS's RR digest type field.

digest a pointer to a buffer containing the DS record's digest data.

dlen the size, in bytes, of the DS record's digest.

ttl the TTL value indicated by the DS resource record.

Returns:

NULL on failure, or a pointer to the new DS object cache entry on success.

Definition at line 796 of file dns.c.

References `_add_cached_object_cmp()`, `_destroy_ds()`, `_ds_comparator()`, `_fixup_ds_links()`, `_get_cache_obj_data()`, `ds::algorithm`, `cached_data_ds`, `cached_stores`, `ds::digest`, `ds::digest_type`, `ds::diglen`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `ds::keytag`, `ds::label`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_lookup_ds()`.

5.353.2.4 void* _clone_dnskey_record_cb (void *record)

Definition at line 2384 of file dns.c.

References `_destroy_dnskey()`, `_ptr_chain_clone()`, `dnskey::algorithm`, `dnskey::dse`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `dnskey::pubkey`, `PUSH_ERROR_SYSCALL`, `dnskey::rdata`, `dnskey::rdlen`, `RET_ERROR_PTR`, `RSAPublicKey_dup_d`, `dnskey::signkeys`, and `dnskey::validated`.

5.353.2.5 int _compare_rdata (const unsigned char *rdata1, size_t rdlen1, const unsigned char *rdata2, size_t rdlen2)

Compare two RR's, for the purpose of allowing for canonical ordering within an RRSET.

Note:

This function is used to sort resource records for the reliable validation of RRSIG records.

According to Section 6.3 of RFC 4034 ("Canonical RR Ordering within an RRSET"): This is treated as a left-justified octet sequence where the absence of an octet sorts before a zero octet.

Parameters:

rdata1 a pointer to the beginning of the first resource record's rdata.

rdlen1 the length, in bytes, of the first resource record's rdata.

rdata2 a pointer to the beginning of the second resource record's rdata.

rdlen2 the length, in bytes, of the second resource record's rdata.

Returns:

-1 if `rdata1 < rdata2`, 1 if `rdata2 > rdata1`, or 0 if the two resource records were equal.

Definition at line 2575 of file dns.c.

Referenced by `_sort_rrs_canonical()`.

5.353.2.6 int _compute_dnskey_sha_hash (const dnskey_t *key, size_t nbits, unsigned char *outbuf)

Perform a SHA-family hash against a DNSKEY public key.

See also:

`compute_sha_hash()`

Note:

This hash value is the data that is used to link a DNSKEY to its corresponding DS record.

According to section 5.1.4 of RFC 4034, the digest is taken over the concatenation of: the owner name of the DNSKEY record in canonical form, with the DNSKEY rdata appended.

Parameters:

- key* a pointer to the DNSKEY record to be hashed.
- nbits* the number of bits to be used for the SHA hash (160, 256, or 512).
- outbuf* a pointer to the output buffer of size nbits bits to receive the DNSKEY hash value.

Returns:

- 1 on general error or 0 on success.

Definition at line 654 of file dns.c.

References `_compute_sha_hash()`, `_mem_append()`, `_mem_append_canon()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dnskey::label`, `dnskey::rdata`, `dnskey::rdlen`, and `RET_ERROR_INT`.

Referenced by `_dump_dnskey_record_cb()`, `_get_ds_by_dnskey()`, and `_lookup_ds()`.

5.353.2.7 void* _deserialize_dnskey_record_cb (void * data, size_t len)

A callback handler to deserialize a DNSKEY record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

- data* a pointer to a buffer containing the data to be deserialized.
- len* the length, in bytes, of the data buffer to be deserialized.

Returns:

- a pointer to a newly allocated `dnskey_t` structure on success, or `NULL` on failure.

Definition at line 2264 of file dns.c.

References `_decode_rsa_pubkey()`, `_deserialize_data()`, `_deserialize_string()`, `_deserialize_vardata()`, `_destroy_dnskey()`, `dnskey::algorithm`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `dnskey::pubkey`, `PUSH_ERROR_SYSCALL`, `dnskey::rdata`, `dnskey::rdlen`, `RET_ERROR_PTR`, and `dnskey::validated`.

5.353.2.8 void* _deserialize_ds_record_cb (void * data, size_t len)

A callback handler to deserialize a DS record from the object cache.

Definition at line 2198 of file dns.c.

References `_deserialize_data()`, `_deserialize_string()`, `_deserialize_vardata()`, `_destroy_ds()`, `ds::algorithm`, `ds::digest`, `ds::digest_type`, `ds::diglen`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `ds::keytag`, `ds::label`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, and `ds::validated`.

5.353.2.9 void _destroy_dnskey (dnskey_t * key)

Free a DNSKEY object and its underlying data.

Parameters:

- key* a pointer to the DNSKEY object to be destroyed.

Definition at line 1071 of file dns.c.

References `dnskey::dse`, `dnskey::label`, `dnskey::pubkey`, `dnskey::rdata`, `RSA_free_d`, and `dnskey::signkeys`.

Referenced by `_add_dnskey_entry_rsa()`, `_clone_dnskey_record_cb()`, `_deserialize_dnskey_record_cb()`, and `_destroy_dnskey_record_cb()`.

5.353.2.10 void _destroy_dnskey_record_cb (void * *record*)

A callback handler to destroy a DNSKEY record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the DNSKEY structure to be destroyed.

Definition at line 2033 of file dns.c.

References _destroy_dnskey().

5.353.2.11 void _destroy_ds (ds_t * *ds*)

Free a DS object and its underlying data.

Parameters:

ds a pointer to the DS object to be destroyed.

Definition at line 1108 of file dns.c.

References ds::digest, ds::label, and ds::signkeys.

Referenced by _add_ds_entry(), _deserialize_ds_record_cb(), and _destroy_ds_record_cb().

5.353.2.12 void _destroy_ds_record_cb (void * *record*)

A callback handler to destroy a DS record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the DS structure to be destroyed.

Definition at line 2148 of file dns.c.

References _destroy_ds().

5.353.2.13 int _dnskey_domain_comparator (const void * *object*, const void * *key*)

An object cache callback for comparing two DNSKEY records based on domain label.

Returns:

-1 if $a < b$, 1 if $b > a$, or 0 if the records were equivalent.

Definition at line 996 of file dns.c.

References dnskey::label.

Referenced by _load_dnskey_file().

5.353.2.14 int _dnskey_tag_comparator (const void * *object*, const void * *key*)

An object cache callback for comparing two DNSKEY records based on tag value.

Note:

This function also checks to make sure that the domain labels match as well.

Parameters:

object a pointer to the DNSKEY cache entry to be compared against the key.

key a pointer to a key value to be compared against the object.

Returns:

-1 if $a < b$, 1 if $b > a$, or 0 if the records were equivalent.

Definition at line 962 of file dns.c.

References dnskey::keytag, and dnskey::label.

Referenced by _add_dnskey_entry_rsa(), and _get_dnskey_by_tag().

5.353.2.15 int _ds_comparator (const void * *object*, const void * *key*)

An object cache callback for comparing two DS records for general equivalence.

Note:

This function is used to map a DNSKEY record to a DS record.

Definition at line 1021 of file dns.c.

References ds::digest, ds::diglen, ds::keytag, and ds::label.

Referenced by _add_ds_entry(), and _get_ds_by_dnskey().

5.353.2.16 void _dump_dns_header (ns_msg * *handle*)

Dump the contents of a DNS reply header.

Parameters:

handle a pointer to the DNS reply to have its header dumped to the console.

Definition at line 1297 of file dns.c.

References _dbgprint().

Referenced by _get_mx_records(), _get_txt_record(), _lookup_dnskey(), and _lookup_ds().

5.353.2.17 void _dump_dnskey_record_cb (FILE * *fp*, void * *record*, int *brief*)

A callback handler to dump a DNSKEY record to the console.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

fp a pointer to the file stream that will receive the dump output.

record a pointer to the DNSKEY structure to be dumped.

brief if set, only print a brief one-line description for the requested record.

Definition at line 2046 of file dns.c.

References `_clear_error_stack()`, `_compute_dnskey_sha_hash()`, `_count_ptr_chain()`, `_dump_buf()`, `_get_signing_key_names()`, `_is_validated_key()`, `dnskey::algorithm`, `DNSSEC_DIGEST_SHA1`, `DNSSEC_DIGEST_SHA256`, `dnskey::dse`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `NS_ALG_RSASHA1`, `NS_ALG_RSASHA256`, `NS_ALG_RSASHA512`, `dnskey::rdlen`, `SHA_160_SIZE`, `SHA_256_SIZE`, `SHA_512_SIZE`, and `dnskey::signkeys`.

5.353.2.18 void _dump_ds_record_cb (FILE *fp, void *record, int brief)

A callback handler to dump a DS record to a file.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

fp the file to which the DS record is dumped.

record a pointer to the DS structure to be dumped.

brief if set, only print a brief one-line description for the requested record.

Definition at line 2161 of file dns.c.

References `_dump_buf()`, `_get_signing_key_names()`, `ds::algorithm`, `ds::digest`, `ds::digest_type`, `ds::diglen`, `ds::keytag`, `ds::label`, `ds::signkeys`, and `ds::validated`.

5.353.2.19 void _fixup_dnskey_validation (void)

Mark as validated any DNSKEY entries in the object cache that should be but aren't.

Note:

The retrieval of a new, validated DNSKEY can mean that another DNSKEY in the cache that was signed by it will now be validated by virtue of transitivity.

Definition at line 2470 of file dns.c.

References `_is_validated_key()`, `_lock_cache_store()`, `_unlock_cache_store()`, `cached_data_dnskey`, `cached_stores`, `cached_object::data`, `dnskey::do_cache`, `cached_store_t::head`, `cached_object::next`, `cached_object::persists`, `store`, and `dnskey::validated`.

Referenced by `_add_dnskey_entry()`, and `_get_txt_record()`.

5.353.2.20 void _fixup_ds_links (void)

Link any DNSKEY entries in the cache with dangling DS pointers to the right place.

Note:

This function should be used after the retrieval of new records, or after a cache load.

Definition at line 2442 of file dns.c.

Referenced by `_add_ds_entry()`.

5.353.2.21 void _free_mx_records (mx_record_t ** *mxs*)

Free a collection of MX records returned by [_get_mx_records\(\)](#).

Parameters:

mxs the array of MX records to be freed.

Definition at line 1837 of file dns.c.

5.353.2.22 dnskey_t* _get_dnskey_by_tag (unsigned int *tag*, const char * *signer*, int *force_lookup*)

Find a DNSKEY entry by its keytag value.

Parameters:

tag the semi-unique keytag value identifying the target DNSKEY entry.

signer a null-terminated string containing the name of the signing domain that owns the DNSKEY entry.

force_lookup if set, perform a live-lookup of any DNSKEY entry that could not be located by keytag; otherwise, lookups will be restricted only to the object cache.

Returns:

a pointer to the specified DNSKEY record if it was found, or NULL on error or if it wasn't.

Definition at line 868 of file dns.c.

References [_clear_error_stack\(\)](#), [_dbgprint\(\)](#), [_dnskey_tag_comparator\(\)](#), [_find_cached_object_cmp\(\)](#), [_get_cache_obj_data\(\)](#), [_lookup_dnskey\(\)](#), [_lookup_ds\(\)](#), [cached_data_dnskey](#), [cached_stores](#), [ERR_BAD_PARAM](#), [IS_ROOT_LABEL](#), [dnskey::keytag](#), [dnskey::label](#), and [RET_ERROR_PTR](#).

Referenced by [_lookup_ds\(\)](#), and [_validate_rrsig_rr\(\)](#).

5.353.2.23 ds_t* _get_ds_by_dnskey (const dnskey_t * *key*)

Look up a DS record by its matching DNSKEY record.

Parameters:

key a pointer to the DNSKEY record that will be used to find the matching DS record.

Returns:

a pointer to the matching DS record on success or NULL on failure.

Definition at line 912 of file dns.c.

References [_clear_error_stack\(\)](#), [_compute_dnskey_sha_hash\(\)](#), [_ds_comparator\(\)](#), [_find_cached_object_cmp\(\)](#), [_get_cache_obj_data\(\)](#), [dnskey::algorithm](#), [ds::algorithm](#), [cached_data_ds](#), [cached_stores](#), [ds::digest](#), [ds::digest_type](#), [ds::diglen](#), [DNSSEC_DIGEST_SHA1](#), [DNSSEC_DIGEST_SHA256](#), [ERR_BAD_PARAM](#), [ERR_UNSPEC](#), [dnskey::keytag](#), [ds::keytag](#), [dnskey::label](#), [ds::label](#), [NS_ALG_RSASHA1](#), [NS_ALG_RSASHA256](#), [RET_ERROR_PTR](#), [RET_ERROR_PTR_FMT](#), [SHA_160_SIZE](#), and [SHA_256_SIZE](#).

5.353.2.24 unsigned int _get_keytag (const unsigned char * *rdata*, size_t *rdlen*)

Calculate the keytag value for a DNSKEY resource record.

Note:

This value is not necessarily unique, but should distinguish between keys of different owners/algorithms. As per RFC 4034, this applies to all algorithms except of type 1.

Parameters:

rdata a pointer to the start of the rdata buffer of the DNSKEY RR.

rdlen the rdlength (total size, in bytes) of the DNSKEY RR.

Returns:

a keytag value for the specified DNSKEY RR to be referenced in RRSIG (or DS) records.

Definition at line 105 of file dns.c.

References ERR_BAD_PARAM, and RET_ERROR_UINT.

Referenced by _add_dnskey_entry(), and _load_dnskey_file().

5.353.2.25 mx_record_t _get_mx_records (const char * *qstring*)**

Retrieve the collection of MX records for a given domain.

Parameters:

qstring a null-terminated string containing the domain name to be queried via DNS.

Returns:

NULL on failure, or a pointer to a null-entry terminated array of mx_record pointers describing the domain on success.

Definition at line 1857 of file dns.c.

References _dbgprint(), _dump_dns_header(), _mem_append(), ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, INITIALIZE_DNS, PUSH_ERROR_RESOLVER, PUSH_ERROR_SYSCALL, RET_ERROR_PTR, and RET_ERROR_PTR_FMT.

Referenced by _sgnt_resolv_dmtip_connect().

5.353.2.26 RSA* _get_rsa_dnskey (const unsigned char * *rdptr*, size_t *rdlen*)

Extract an RSA public key from a DNSKEY resource record.

Note:

The parameters to this function are not the same as the RR's rdata/rdlen, but rather, the RSA key data that usually begins 4 bytes into this buffer.

Parameters:

rdptr a pointer to the start of the raw RSA key data inside the DNSKEY RR's rdata.

rdlen the length, in bytes, of the encoded RSA public key.

Returns:

NULL on failure, or a pointer to a newly allocated RSA public key on success.

Definition at line 132 of file dns.c.

References _dbgprint(), BN_bin2bn_d, BN_free_d, BN_num_bits_d, ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_OPENSSL, RET_ERROR_PTR, RET_ERROR_PTR_FMT, and RSA_new_d.

Referenced by _add_dnskey_entry(), and _load_dnskey_file().

5.353.2.27 char* _get_signing_key_names (dnskey_t ** keys)

Get the names of a collection of signing DNSKEYs for human display.

Parameters:

keys a pointer to an array of DNSKEY entries to be printed to a human-readable string.

Returns:

NULL on failure, or a pointer to a newly allocated null-terminated string containing the DNSKEY collection info on success.

Definition at line 2613 of file dns.c.

References `_str_printf()`, `ERR_NOMEM`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_dump_dnskey_record_cb()`, and `_dump_ds_record_cb()`.

5.353.2.28 char* _get_txt_record (const char * qstring, unsigned long * ttl, int * validated)

Get the answer to a DNS TXT record query.

Parameters:

qstring a pointer to a null-terminated string containing the DNS query string.

ttl if not NULL, an optional pointer to a value that will store the TTL of the retrieved record on success.

validated

Definition at line 1598 of file dns.c.

References `_clear_error_stack()`, `_dbgprint()`, `_dump_buf()`, `_dump_dns_header()`, `_fixup_dnskey_validation()`, `_is_validated_key()`, `_validate_rrsig_rr()`, `_verbose`, `dump_error_stack()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `INITIALIZE_DNS`, `IS_ROOT_LABEL`, `PUSH_ERROR_RESOLVER`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, `RET_ERROR_PTR_FMT`, and `T_RRSIG`.

Referenced by `_get_dime_record()`.

5.353.2.29 int _initialize_resolver (void)

Initialize the DNS resolver subsystem.

Returns:

-1 on failure or 0 on success.

Definition at line 1994 of file dns.c.

References `_dbgprint()`, `_get_dime_dir_location()`, `_load_dnskey_file()`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, `RET_ERROR_INT`, and `ROOT_KEY_FILE`.

5.353.2.30 int _is_validated_key (dnskey_t * dk)

Trace a DNSKEY backwards through its DS entry to the root, to see if it is validated.

Note:

Chain validation works recursively on a starting DNSKEY in the following way: 1. For each of the DNSKEYs that were used to sign this DNSKEY record, 2. For each of the signing DNSKEY's DS records, 3. Validate the signing DNSKEY of the DS record. In the end, a DNSKEY can only be considered if its validated flag has been set, or a key by which it has been signed has also been validated. Only the keys supplied from the function `_load_dnskey_file()` carry the validation flag by default.

Parameters:

dk a pointer to the DNSKEY to be validated.

Returns:

-1 on general error, 0 if the specified key could not be validated, or 1 if it is.

Definition at line 593 of file dns.c.

References ERR_BAD_PARAM, IS_ROOT_LABEL, dnskey::label, RET_ERROR_INT, dnskey::signkeys, and dnskey::validated.

Referenced by _dump_dnskey_record_cb(), _fixup_dnskey_validation(), and _get_txt_record().

5.353.2.31 int _load_dnskey_file (const char **filename*)

Load a collection of DNS key(s) from a config file (into the object cache).

Note:

Any DNSKEY entry loaded from this file will be treated as verified and having final authority. This function will fail if there are no root [...] DNSKEY entries supplied in the file.

Parameters:

filename a null-terminated string containing the name of the file containing the DNS key entries.

Returns:

0 if key(s) were successfully loaded from the file or -1 on failure.

Definition at line 424 of file dns.c.

References _add_dnskey_entry_rsa(), _b64decode(), _dnskey_domain_comparator(), _find_cached_object_cmp(), _get_keytag(), _get_rsa_dnskey(), _mem_append(), cached_data_dnskey, cached_stores, chr_isspace, DNSKEY_RR_FLAG_SEP, DNSKEY_RR_FLAG_ZK, DNSKEY_RR_PROTO, ERR_BAD_PARAM, ERR_UNSPEC, dnskey::label, NS_ALG_RSASHA1, NS_ALG_RSASHA256, NS_ALG_RSASHA512, PUSH_ERROR_SYSCALL, RET_ERROR_INT, RET_ERROR_INT_FMT, and dnskey::validated.

Referenced by _initialize_resolver().

5.353.2.32 void* _lookup_dnskey (const char **label*)

Definition at line 1309 of file dns.c.

References _add_dnskey_entry(), _clear_error_stack(), _dbgprint(), _dump_dns_header(), _ptr_chain_add(), _validate_rrsig_rr(), dump_error_stack(), ERR_UNSPEC, PUSH_ERROR_RESOLVER, RET_ERROR_PTR, T_DNSKEY, and T_RRSIG.

Referenced by _get_dnskey_by_tag().

5.353.2.33 void* _lookup_ds (const char **label*)

Lookup the DS record for a specified domain.

Parameters:

label a null-terminated string containing the name of the domain to have its DS record queried.

Definition at line 1422 of file dns.c.

References _add_ds_entry(), _clear_error_stack(), _compute_dnskey_sha_hash(), _dbgprint(), _dump_dns_header(), _get_dnskey_by_tag(), _ptr_chain_add(), _validate_rrsig_rr(), DNSSEC_DIGEST_SHA1, DNSSEC_DIGEST_SHA256, ds_rr_t, dnskey::dse, dump_error_stack(), ERR_BAD_PARAM, ERR_UNSPEC, NS_ALG_RSASHA1, NS_ALG_RSASHA256, NS_ALG_RSASHA512, PUSH_ERROR_RESOLVER, RET_ERROR_PTR, T_DS, and T_RRSIG.

Referenced by `_get_dnskey_by_tag()`.

5.353.2.34 `size_t _mem_append_canon (unsigned char ** buf, size_t * blen, const char * name)`

Append a DNS label in uncompressed, canonical format to a dynamic buffer.

Note:

This function should be called for the first time with **buf* and **blen* set to NULL.

Parameters:

buf a pointer to the address of the output buffer that will be resized to hold the result, or NULL to allocate one.

blen a pointer to a variable that will both hold the initial size of and receive the newly resized output buffer following the completion of the append operation.

name a null-terminated string containing the domain label to be stored in the buffer in canonical form.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 23 of file `dns.c`.

References `_mem_append()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_UINT`.

Referenced by `_compute_dnskey_sha_hash()`, and `_rsa_verify_record()`.

5.353.2.35 `int _rsa_verify_record (const char * label, unsigned char algorithm, RSA * pubkey, const unsigned char * rrsig, const unsigned char * sigbuf, size_t siglen, ns_msg * dhandle)`

Verify that an RRSIG record has been signed properly using a given RSA public key.

Note:

According to RFC 4034, the signature calculation is computed over the following data: `RRSIG_RDATA | RR(1) | RR(2) | RR(s)` where `RRSIG_RDATA` is the wire format of the RRSIG RDATA with the signer's name in canonical form and NO signature field `RR(s)` is the RRset of all matching RR's. `RR(x): owner | type | class | ttl (original ttl) | rdlen | rdata`

These fields of each RR and the RRSIG RR must match: owner, class, type=covered, ttl=original ttl The RRSET must be in canonical order, and the DNS names in the RDATA must be in canonical form.

Each domain name in an RR must be fully expanded and qualified, and in lowercase. Finally, the RR's need to be sorted in canonical order.

Parameters:

label a pointer to the label of the RRSIG record being verified.

algorithm the value of the algorithm used to produce the signature (ex. `NS_ALG_RSASHA1`).

pubkey a pointer to the RSA public key that will be used to verify the specified signature.

rrsig a pointer to the beginning of the RRSIG record's rdata content.

sigbuf a pointer to the start of the input buffer containing the RR signature data.

siglen the length, in bytes, of the RR signature data.

dhandle a pointer to the handle of the DNS answer message containing the RRSIG record being verified.

Returns:

-1 on general error, 0 if the RSA signature value did not match the data, or 1 if the signature was valid.

Definition at line 233 of file dns.c.

References `_dbgprint()`, `_mem_append()`, `_mem_append_canon()`, `_sort_rrs_canonical()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `EVP_DigestInit_d`, `EVP_DigestUpdate_d`, `EVP_MD_CTX_init_d`, `EVP_PKEY_new_d`, `EVP_PKEY_set1_RSA_d`, `EVP_sha1_d`, `EVP_sha256_d`, `EVP_sha512_d`, `EVP_VerifyFinal_d`, `NS_ALG_RSASHA1`, `NS_ALG_RSASHA256`, `NS_ALG_RSASHA512`, `PUSH_ERROR_OPENSSL`, `PUSH_ERROR_RESOLVER`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, `RET_ERROR_INT_FMT`, and `rrsig_rr_t`.

Referenced by `_validate_rrsig_rr()`.

5.353.2.36 `void* _serialize_dnskey_record_cb (void *record, size_t *outlen)`

A callback handler to serialize a DNSKEY record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the DNSKEY structure to be serialized.

outlen a pointer to a variable that will receive the length of the serialized DNSKEY.

Returns:

a pointer to a newly allocated buffer holding the serialized DNSKEY on success, or NULL on failure.

Definition at line 2341 of file dns.c.

References `_encode_rsa_pubkey()`, `_mem_append()`, `_mem_append_serialized()`, `_mem_append_serialized_string()`, `dnskey::algorithm`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `dnskey::pubkey`, `dnskey::rdata`, `dnskey::rdlen`, `RET_ERROR_PTR`, and `dnskey::validated`.

5.353.2.37 `void* _serialize_ds_record_cb (void *record, size_t *outlen)`

A callback record to serialize a DS record for storage in the object cache.

Definition at line 2233 of file dns.c.

References `_mem_append()`, `_mem_append_serialized()`, `_mem_append_serialized_string()`, `ds::algorithm`, `ds::digest`, `ds::digest_type`, `ds::diglen`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ds::keytag`, `ds::label`, `RET_ERROR_PTR`, and `ds::validated`.

5.353.2.38 `int _sort_rrs_canonical (ns_msg *dhandle, int *ordbuf, size_t nanswers)`

Sort the RRs in an RRSET into canonical order for DNSSEC signature verification.

Note:

This function does not actually sort the targeted RR's, but rather produces an output buffer containing an array of the sorted RR indices.

Parameters:

dhandle the handle of the DNS reply containing the answer RRs to be sorted.

ordbuf a pointer to an array of integers of size nanswers that will be updated to contain the new zero-indexed RR indices in proper canonically sorted order.

nanswers the number of answers in the DNS reply to be sorted (must match dhandle).

Returns:

-1 on error or 0 on success.

Definition at line 2518 of file dns.c.

References `_compare_rdata()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_RESOLVER`, and `RET_ERROR_INT`.

Referenced by `_rsa_verify_record()`.

5.353.2.39 `int _validate_rrsig_rr (const char * label, ns_msg * dhandle, unsigned short covered, const unsigned char * rdata, size_t rdlen, dnskey_t ** outkey)`

Verify the signature provided by an RRSIG record.

See also:

`rsa_verify_record()`

Returns:

1 if the RRSIG signature was correct for the RR data, 0 if it was not, or -1 on general error.

Definition at line 1204 of file dns.c.

References `_dbgprint()`, `_dump_buf_outer()`, `_get_chr_date()`, `_get_dnskey_by_tag()`, `_rsa_verify_record()`, `_verbose`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `NS_ALG_RSASHA1`, `NS_ALG_RSASHA256`, `NS_ALG_RSASHA512`, `dnskey::pubkey`, `PUSH_ERROR_RESOLVER`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_INT`, `RET_ERROR_INT_FMT`, and `rrsig_rr_t`.

Referenced by `_get_txt_record()`, `_lookup_dnskey()`, and `_lookup_ds()`.

5.354 src/providers/dime/signet-resolver/dns.h File Reference

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser_compat.h>
#include <netdb.h>
#include <resolv.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <openssl/ssl.h>
#include <openssl/rsa.h>
#include <openssl/err.h>
#include "dime/common/error.h"
```

Data Structures

- struct [ds](#)
- struct [dnskey](#)
- struct [mx_record_t](#)

Defines

- #define [ROOT_KEY_FILE](#) "root-anchor.key"
- #define [IS_ROOT_LABEL](#)(lname) (!lname || !strlen(lname) || *lname == '.')
- #define [T_DNSKEY](#) 48
- #define [T_DS](#) 43
- #define [T_RRSIG](#) 46
- #define [DNSKEY_RR_LEN](#) 4
- #define [DNSKEY_RR_FLAG_ZK](#) 0x100
- #define [DNSKEY_RR_FLAG_SEP](#) 0x001
- #define [DNSKEY_RR_PROTO](#) 3
- #define [NS_ALG_RSASHA1](#) 5
- #define [NS_ALG_RSASHA256](#) 8
- #define [NS_ALG_RSASHA512](#) 10
- #define [DNSSEC_DIGEST_SHA1](#) 1
- #define [DNSSEC_DIGEST_SHA256](#) 2

Typedefs

- typedef struct [ds](#) [ds_t](#)
- typedef struct [dnskey](#) [dnskey_t](#)

Functions

- struct `__attribute__((packed))`
- `PUBLIC_FUNC_DECL` (int, load_dnskey_file, const char *filename)
- `PUBLIC_FUNC_DECL` (int, is_validated_key, `dnskey_t` *dk)
- `PUBLIC_FUNC_DECL` (int, compute_dnskey_sha_hash, const `dnskey_t` *key, size_t nbits, unsigned char *outbuf)
- `PUBLIC_FUNC_DECL` (`dnskey_t` *, get_dnskey_by_tag, unsigned int tag, const char *signer, int force_lookup)
- `PUBLIC_FUNC_DECL` (`ds_t` *, get_ds_by_dnskey, const `dnskey_t` *key)
- `PUBLIC_FUNC_DECL` (unsigned int, get_keytag, const unsigned char *rdata, size_t rdlen)
- `PUBLIC_FUNC_DECL` (void, destroy_dnskey, `dnskey_t` *key)
- `PUBLIC_FUNC_DECL` (void, destroy_ds, `ds_t` *ds)
- `PUBLIC_FUNC_DECL` (int, rsa_verify_record, const char *label, unsigned char algorithm, RSA *pubkey, const unsigned char *rrsig, const unsigned char *sigbuf, size_t siglen, ns_msg *dhandle)
- `PUBLIC_FUNC_DECL` (int, validate_rrsig_rr, const char *label, ns_msg *dhandle, unsigned short covered, const unsigned char *rdata, size_t rdlen, `dnskey_t` **outkey)
- `PUBLIC_FUNC_DECL` (void *, lookup_dnskey, const char *label)
- `PUBLIC_FUNC_DECL` (void *, lookup_ds, const char *label)
- `PUBLIC_FUNC_DECL` (char *, get_txt_record, const char *qstring, unsigned long *ttl, int *validated)
- `PUBLIC_FUNC_DECL` (`mx_record_t` **, get_mx_records, const char *qstring)
- `PUBLIC_FUNC_DECL` (void, free_mx_records, `mx_record_t` **mxs)
- int `_initialize_resolver` (void)
Initialize the DNS resolver subsystem.
- `dnskey_t` * `_add_dnskey_entry` (const char *label, const unsigned char *buf, size_t len, unsigned long ttl)
Create and add a new DNSKEY entry to the object cache.
- `dnskey_t` * `_add_dnskey_entry_rsa` (const char *label, uint16_t flags, unsigned char algorithm, RSA *pubkey, unsigned int keytag, const unsigned char *rdata, size_t rdlen, unsigned long ttl, unsigned int do_cache, int forced)
Create and add a new RSA DNSKEY entry to the object cache.
- `ds_t` * `_add_ds_entry` (const char *label, unsigned int keytag, unsigned char algorithm, unsigned char digest_type, const unsigned char *digest, size_t dlen, unsigned long ttl)
Create and add a new DS entry to the object cache.
- RSA * `_get_rsa_dnskey` (const unsigned char *rdptr, size_t rdlen)
Extract an RSA public key from a DNSKEY resource record.
- void `_destroy_dnskey_record_cb` (void *record)
A callback handler to destroy a DNSKEY record.
- void `_destroy_ds_record_cb` (void *record)
A callback handler to destroy a DS record.
- void * `_serialize_dnskey_record_cb` (void *record, size_t *outlen)
A callback handler to serialize a DNSKEY record.
- void * `_serialize_ds_record_cb` (void *record, size_t *outlen)
A callback record to serialize a DS record for storage in the object cache.
- void * `_deserialize_dnskey_record_cb` (void *data, size_t len)
A callback handler to deserialize a DNSKEY record.
- void * `_deserialize_ds_record_cb` (void *data, size_t len)

A callback handler to deserialize a DS record from the object cache.

- void * [_clone_dnskey_record_cb](#) (void *record)
- void [_dump_dns_header](#) (ns_msg *handle)
Dump the contents of a DNS reply header.
- void [_dump_dnskey_record_cb](#) (FILE *fp, void *record, int brief)
A callback handler to dump a DNSKEY record to the console.
- void [_dump_ds_record_cb](#) (FILE *fp, void *record, int brief)
A callback handler to dump a DS record to a file.
- void [_fixup_ds_links](#) (void)
Link any DNSKEY entries in the cache with dangling DS pointers to the right place.
- void [_fixup_dnskey_validation](#) (void)
Mark as validated any DNSKEY entries in the object cache that should be but aren't.
- int [_sort_rrs_canonical](#) (ns_msg *dhandle, int *ordbuf, size_t nanswers)
Sort the RRs in an RRSET into canonical order for DNSSEC signature verification.
- int [_compare_rdata](#) (const unsigned char *rdata1, size_t rrlen1, const unsigned char *rdata2, size_t rrlen2)
Compare two RR's, for the purpose of allowing for canonical ordering within an RRSET.
- char * [_get_signing_key_names](#) (dnskey_t **keys)
Get the names of a collection of signing DNSKEYs for human display.
- int [_ds_comparator](#) (const void *object, const void *key)
An object cache callback for comparing two DS records for general equivalence.
- int [_dnskey_tag_comparator](#) (const void *object, const void *key)
An object cache callback for comparing two DNSKEY records based on tag value.
- int [_dnskey_domain_comparator](#) (const void *object, const void *key)
An object cache callback for comparing two DNSKEY records based on domain label.
- size_t [_mem_append_canon](#) (unsigned char **buf, size_t *blen, const char *name)
Append a DNS label in uncompressed, canonical format to a dynamic buffer.

Variables

- [dnskey_rr_flags_t](#)
- [rrsig_rr_t](#)
- [ds_rr_t](#)

5.354.1 Define Documentation

5.354.1.1 #define DNSKEY_RR_FLAG_SEP 0x001

Definition at line 45 of file dns.h.

Referenced by [_add_dnskey_entry_rsa\(\)](#), and [_load_dnskey_file\(\)](#).

5.354.1.2 #define DNSKEY_RR_FLAG_ZK 0x100

Definition at line 44 of file dns.h.

Referenced by `_add_dnskey_entry_rsa()`, and `_load_dnskey_file()`.

5.354.1.3 #define DNSKEY_RR_LEN 4

Definition at line 43 of file dns.h.

5.354.1.4 #define DNSKEY_RR_PROTO 3

Definition at line 46 of file dns.h.

Referenced by `_add_dnskey_entry()`, and `_load_dnskey_file()`.

5.354.1.5 #define DNSSEC_DIGEST_SHA1 1

Definition at line 52 of file dns.h.

Referenced by `_dump_dnskey_record_cb()`, `_get_ds_by_dnskey()`, and `_lookup_ds()`.

5.354.1.6 #define DNSSEC_DIGEST_SHA256 2

Definition at line 53 of file dns.h.

Referenced by `_dump_dnskey_record_cb()`, `_get_ds_by_dnskey()`, and `_lookup_ds()`.

5.354.1.7 #define IS_ROOT_LABEL(lname) (!lname || !strlen(lname) || *lname == '.')

Definition at line 27 of file dns.h.

Referenced by `_get_dnskey_by_tag()`, `_get_txt_record()`, and `_is_validated_key()`.

5.354.1.8 #define NS_ALG_RSASHA1 5

Definition at line 48 of file dns.h.

Referenced by `_add_dnskey_entry()`, `_dump_dnskey_record_cb()`, `_get_ds_by_dnskey()`, `_load_dnskey_file()`, `_lookup_ds()`, `_rsa_verify_record()`, and `_validate_rrsig_rr()`.

5.354.1.9 #define NS_ALG_RSASHA256 8

Definition at line 49 of file dns.h.

Referenced by `_add_dnskey_entry()`, `_dump_dnskey_record_cb()`, `_get_ds_by_dnskey()`, `_load_dnskey_file()`, `_lookup_ds()`, `_rsa_verify_record()`, and `_validate_rrsig_rr()`.

5.354.1.10 #define NS_ALG_RSASHA512 10

Definition at line 50 of file dns.h.

Referenced by `_add_dnskey_entry()`, `_dump_dnskey_record_cb()`, `_load_dnskey_file()`, `_lookup_ds()`, `_rsa_verify_record()`, and `_validate_rrsig_rr()`.

5.354.1.11 #define ROOT_KEY_FILE "root-anchor.key"

Definition at line 25 of file dns.h.

Referenced by `_initialize_resolver()`.

5.354.1.12 #define T_DNSKEY 48

Definition at line 32 of file dns.h.

Referenced by `_lookup_dnskey()`.

5.354.1.13 #define T_DS 43

Definition at line 36 of file dns.h.

Referenced by `_lookup_ds()`.

5.354.1.14 #define T_RRSIG 46

Definition at line 40 of file dns.h.

Referenced by `_get_txt_record()`, `_lookup_dnskey()`, and `_lookup_ds()`.

5.354.2 Typedef Documentation**5.354.2.1 typedef struct dnskey dnskey_t**

Definition at line 91 of file dns.h.

5.354.2.2 typedef struct ds ds_t

Definition at line 90 of file dns.h.

5.354.3 Function Documentation**5.354.3.1 struct __attribute__((packed)) [read]**

- < The type of the rr set covered by this record.
- < The numerical id of the cryptographic algorithm used to generate the signature.
- < The number of labels in the original RRSIG RR owner name.
- < The original TTL of the covered rrset as it appears in the authoritative zone.
- < The expiration date after which the RRSIG record must not be used for authentication.
- < The inception date before which the RRSIG record must not be used for authentication.
- < A tag (selector) identifying the corresponding DNSKEY RR (which isn't necessarily unique) that validates this record.
- < A label identifying the owner name of the DNSKEY RR validating this signature (must not use name compression).
- < The keytag of the DNSKEY RR to which this record refers.
- < The algorithm used by this RR's referenced DNSKEY.
- < The type of the digest algorithm used by the digest field.

< A digest of this RR's referenced DNSKEY.

Definition at line 82 of file dns.h.

5.354.3.2 `dnskey_t* _add_dnskey_entry (const char * label, const unsigned char * buf, size_t len, unsigned long ttl)`

Create and add a new DNSKEY entry to the object cache.

Parameters:

label a pointer to a null-terminated string containing the name of the domain supplying the specified key.

buf a pointer to the beginning of the DNSKEY's RR rdata.

len the length, in bytes, of the DNSKEY RR's rdata.

ttl the TTL value indicated by the DNSKEY resource record.

Definition at line 1140 of file dns.c.

References `_add_dnskey_entry_rsa()`, `_dbgprint()`, `_fixup_dnskey_validation()`, `_get_keytag()`, `_get_rsa_dnskey()`, `dnskey_rr_flags_t`, `DNSKEY_RR_PROTO`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `NS_ALG_RSASHA1`, `NS_ALG_RSASHA256`, `NS_ALG_RSASHA512`, `RET_ERROR_PTR`, and `RET_ERROR_PTR_FMT`.

Referenced by `_lookup_dnskey()`.

5.354.3.3 `dnskey_t* _add_dnskey_entry_rsa (const char * label, uint16_t flags, unsigned char algorithm, RSA * pubkey, unsigned int keytag, const unsigned char * rdata, size_t rdlen, unsigned long ttl, unsigned int do_cache, int forced)`

Create and add a new RSA DNSKEY entry to the object cache.

Parameters:

label a pointer to a null-terminated string containing the name of the domain supplying the specified key.

flags a bitmask of flags corresponding to the DNSKEY's RR flags field.

algorithm the value of the DNSKEY's RR algorithm field.

pubkey a pointer to an RSA public key used to validate signatures made by this key.

keytag a numerical value linking this to other RRSIG or DS records.

rdata a pointer to the beginning of this DNSKEY's RR rdata.

rdlen the length, in bytes, of the DNSKEY's RR rdata.

ttl the TTL value indicated by the DNSKEY resource record.

do_cache if set, the new object cache entry for this DNSKEY can be persisted to the cache file.

forced if set, this entry will forcibly replace (overshadow) any clashing or matching existing DNSKEY entry inside the object cache.

Returns:

NULL on failure, or a pointer to the new DNSKEY object cache entry on success.

Definition at line 698 of file dns.c.

References `_add_cached_object_cmp()`, `_add_cached_object_cmp_forced()`, `_destroy_dnskey()`, `_dnskey_tag_comparator()`, `_get_cache_obj_data()`, `dnskey::algorithm`, `cached_data_dnskey`, `cached_stores`, `DNSKEY_RR_FLAG_SEP`, `DNSKEY_RR_FLAG_ZK`, `dnskey::do_cache`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `dnskey::pubkey`, `PUSH_ERROR_SYSCALL`, `dnskey::rdata`, `dnskey::rdlen`, and `RET_ERROR_PTR`.

Referenced by `_add_dnskey_entry()`, and `_load_dnskey_file()`.

5.354.3.4 `ds_t* _add_ds_entry (const char * label, unsigned int keytag, unsigned char algorithm, unsigned char digest_type, const unsigned char * digest, size_t dlen, unsigned long tll)`

Create and add a new DS entry to the object cache.

Parameters:

label a pointer to a null-terminated string containing the name of the domain supplying the specified DS record.

keytag a numerical value identifying the DNSKEY entry linked to this record.

algorithm the value of the DS's RR algorithm field.

digest_type the value of the DS's RR digest type field.

digest a pointer to a buffer containing the DS record's digest data.

dlen the size, in bytes, of the DS record's digest.

tll the TTL value indicated by the DS resource record.

Returns:

NULL on failure, or a pointer to the new DS object cache entry on success.

Definition at line 796 of file dns.c.

References `_add_cached_object_cmp()`, `_destroy_ds()`, `_ds_comparator()`, `_fixup_ds_links()`, `_get_cache_obj_data()`, `ds::algorithm`, `cached_data_ds`, `cached_stores`, `ds::digest`, `ds::digest_type`, `ds::diglen`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `ds::keytag`, `ds::label`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_lookup_ds()`.

5.354.3.5 `void* _clone_dnskey_record_cb (void * record)`

Definition at line 2384 of file dns.c.

References `_destroy_dnskey()`, `_ptr_chain_clone()`, `dnskey::algorithm`, `dnskey::dse`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `dnskey::pubkey`, `PUSH_ERROR_SYSCALL`, `dnskey::rdata`, `dnskey::rdlen`, `RET_ERROR_PTR`, `RSAPublicKey_dup_d`, `dnskey::signkeys`, and `dnskey::validated`.

5.354.3.6 `int _compare_rdata (const unsigned char * rdata1, size_t rdlen1, const unsigned char * rdata2, size_t rdlen2)`

Compare two RR's, for the purpose of allowing for canonical ordering within an RRSET.

Note:

This function is used to sort resource records for the reliable validation of RRSIG records.

According to Section 6.3 of RFC 4034 ("Canonical RR Ordering within an RRSET"): This is treated as a left-justified octet sequence where the absence of an octet sorts before a zero octet.

Parameters:

rdata1 a pointer to the beginning of the first resource record's rdata.

rdlen1 the length, in bytes, of the first resource record's rdata.

rdata2 a pointer to the beginning of the second resource record's rdata.

rdlen2 the length, in bytes, of the second resource record's rdata.

Returns:

-1 if `rdata1 < rdata2`, 1 if `rdata2 > rdata1`, or 0 if the two resource records were equal.

Definition at line 2575 of file dns.c.

Referenced by `_sort_rrs_canonical()`.

5.354.3.7 void* _deserialize_dnskey_record_cb (void * *data*, size_t *len*)

A callback handler to deserialize a DNSKEY record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

data a pointer to a buffer containing the data to be deserialized.

len the length, in bytes, of the data buffer to be deserialized.

Returns:

a pointer to a newly allocated dnskey_t structure on success, or NULL on failure.

Definition at line 2264 of file dns.c.

References _decode_rsa_pubkey(), _deserialize_data(), _deserialize_string(), _deserialize_vardata(), _destroy_dnskey(), dnskey::algorithm, ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, dnskey::is_sep, dnskey::is_zone, dnskey::keytag, dnskey::label, dnskey::pubkey, PUSH_ERROR_SYSCALL, dnskey::rdata, dnskey::rdlen, RET_ERROR_PTR, and dnskey::validated.

5.354.3.8 void* _deserialize_ds_record_cb (void * *data*, size_t *len*)

A callback handler to deserialize a DS record from the object cache.

Definition at line 2198 of file dns.c.

References _deserialize_data(), _deserialize_string(), _deserialize_vardata(), _destroy_ds(), ds::algorithm, ds::digest, ds::digest_type, ds::diglen, ERR_BAD_PARAM, ERR_NOMEM, ERR_UNSPEC, ds::keytag, ds::label, PUSH_ERROR_SYSCALL, RET_ERROR_PTR, and ds::validated.

5.354.3.9 void _destroy_dnskey_record_cb (void * *record*)

A callback handler to destroy a DNSKEY record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the DNSKEY structure to be destroyed.

Definition at line 2033 of file dns.c.

References _destroy_dnskey().

5.354.3.10 void _destroy_ds_record_cb (void * *record*)

A callback handler to destroy a DS record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the DS structure to be destroyed.

Definition at line 2148 of file dns.c.

References `_destroy_ds()`.

5.354.3.11 `int _dnskey_domain_comparator (const void * object, const void * key)`

An object cache callback for comparing two DNSKEY records based on domain label.

Returns:

-1 if $a < b$, 1 if $b > a$, or 0 if the records were equivalent.

Definition at line 996 of file dns.c.

References `dnskey::label`.

Referenced by `_load_dnskey_file()`.

5.354.3.12 `int _dnskey_tag_comparator (const void * object, const void * key)`

An object cache callback for comparing two DNSKEY records based on tag value.

Note:

This function also checks to make sure that the domain labels match as well.

Parameters:

object a pointer to the DNSKEY cache entry to be compared against the key.

key a pointer to a key value to be compared against the object.

Returns:

-1 if $a < b$, 1 if $b > a$, or 0 if the records were equivalent.

Definition at line 962 of file dns.c.

References `dnskey::keytag`, and `dnskey::label`.

Referenced by `_add_dnskey_entry_rsa()`, and `_get_dnskey_by_tag()`.

5.354.3.13 `int _ds_comparator (const void * object, const void * key)`

An object cache callback for comparing two DS records for general equivalence.

Note:

This function is used to map a DNSKEY record to a DS record.

Definition at line 1021 of file dns.c.

References `ds::digest`, `ds::diglen`, `ds::keytag`, and `ds::label`.

Referenced by `_add_ds_entry()`, and `_get_ds_by_dnskey()`.

5.354.3.14 `void _dump_dns_header (ns_msg * handle)`

Dump the contents of a DNS reply header.

Parameters:

handle a pointer to the DNS reply to have its header dumped to the console.

Definition at line 1297 of file dns.c.

References `_dbgprint()`.

Referenced by `_get_mx_records()`, `_get_txt_record()`, `_lookup_dnskey()`, and `_lookup_ds()`.

5.354.3.15 void _dump_dnskey_record_cb (FILE *fp, void *record, int brief)

A callback handler to dump a DNSKEY record to the console.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

fp a pointer to the file stream that will receive the dump output.

record a pointer to the DNSKEY structure to be dumped.

brief if set, only print a brief one-line description for the requested record.

Definition at line 2046 of file dns.c.

References `_clear_error_stack()`, `_compute_dnskey_sha_hash()`, `_count_ptr_chain()`, `_dump_buf()`, `_get_signing_key_names()`, `_is_validated_key()`, `dnskey::algorithm`, `DNSSEC_DIGEST_SHA1`, `DNSSEC_DIGEST_SHA256`, `dnskey::dse`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `NS_ALG_RSASHA1`, `NS_ALG_RSASHA256`, `NS_ALG_RSASHA512`, `dnskey::rdlen`, `SHA_160_SIZE`, `SHA_256_SIZE`, `SHA_512_SIZE`, and `dnskey::signkeys`.

5.354.3.16 void _dump_ds_record_cb (FILE *fp, void *record, int brief)

A callback handler to dump a DS record to a file.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

fp the file to which the DS record is dumped.

record a pointer to the DS structure to be dumped.

brief if set, only print a brief one-line description for the requested record.

Definition at line 2161 of file dns.c.

References `_dump_buf()`, `_get_signing_key_names()`, `ds::algorithm`, `ds::digest`, `ds::digest_type`, `ds::diglen`, `ds::keytag`, `ds::label`, `ds::signkeys`, and `ds::validated`.

5.354.3.17 void _fixup_dnskey_validation (void)

Mark as validated any DNSKEY entries in the object cache that should be but aren't.

Note:

The retrieval of a new, validated DNSKEY can mean that another DNSKEY in the cache that was signed by it will now be validated by virtue of transitivity.

Definition at line 2470 of file dns.c.

References `_is_validated_key()`, `_lock_cache_store()`, `_unlock_cache_store()`, `cached_data_dnskey`, `cached_stores`, `cached_object::data`, `dnskey::do_cache`, `cached_store_t::head`, `cached_object::next`, `cached_object::persists`, `store`, and `dnskey::validated`.

Referenced by `_add_dnskey_entry()`, and `_get_txt_record()`.

5.354.3.18 `void _fixup_ds_links (void)`

Link any DNSKEY entries in the cache with dangling DS pointers to the right place.

Note:

This function should be used after the retrieval of new records, or after a cache load.

Definition at line 2442 of file dns.c.

Referenced by `_add_ds_entry()`.

5.354.3.19 `RSA* _get_rsa_dnskey (const unsigned char * rdptr, size_t rdlen)`

Extract an RSA public key from a DNSKEY resource record.

Note:

The parameters to this function are not the same as the RR's `rdata/rdlen`, but rather, the RSA key data that usually begins 4 bytes into this buffer.

Parameters:

rdptr a pointer to the start of the raw RSA key data inside the DNSKEY RR's `rdata`.

rdlen the length, in bytes, of the encoded RSA public key.

Returns:

NULL on failure, or a pointer to a newly allocated RSA public key on success.

Definition at line 132 of file dns.c.

References `_dbgprint()`, `BN_bin2bn_d`, `BN_free_d`, `BN_num_bits_d`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, `RET_ERROR_PTR_FMT`, and `RSA_new_d`.

Referenced by `_add_dnskey_entry()`, and `_load_dnskey_file()`.

5.354.3.20 `char* _get_signing_key_names (dnskey_t ** keys)`

Get the names of a collection of signing DNSKEYs for human display.

Parameters:

keys a pointer to an array of DNSKEY entries to be printed to a human-readable string.

Returns:

NULL on failure, or a pointer to a newly allocated null-terminated string containing the DNSKEY collection info on success.

Definition at line 2613 of file dns.c.

References `_str_printf()`, `ERR_NOMEM`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

Referenced by `_dump_dnskey_record_cb()`, and `_dump_ds_record_cb()`.

5.354.3.21 `int _initialize_resolver (void)`

Initialize the DNS resolver subsystem.

Returns:

-1 on failure or 0 on success.

Definition at line 1994 of file dns.c.

References `_dbgprint()`, `_get_dime_dir_location()`, `_load_dnskey_file()`, `ERR_UNSPEC`, `PUSH_ERROR_FMT`, `RET_ERROR_INT`, and `ROOT_KEY_FILE`.

5.354.3.22 `size_t _mem_append_canon (unsigned char **buf, size_t *blen, const char *name)`

Append a DNS label in uncompressed, canonical format to a dynamic buffer.

Note:

This function should be called for the first time with `*buf` and `*blen` set to `NULL`.

Parameters:

buf a pointer to the address of the output buffer that will be resized to hold the result, or `NULL` to allocate one.

blen a pointer to a variable that will both hold the initial size of and receive the newly resized output buffer following the completion of the append operation.

name a null-terminated string containing the domain label to be stored in the buffer in canonical form.

Returns:

the new size of the (re)allocated output buffer, or 0 on failure.

Definition at line 23 of file dns.c.

References `_mem_append()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_UINT`.

Referenced by `_compute_dnskey_sha_hash()`, and `_rsa_verify_record()`.

5.354.3.23 `void* _serialize_dnskey_record_cb (void *record, size_t *outlen)`

A callback handler to serialize a DNSKEY record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the DNSKEY structure to be serialized.

outlen a pointer to a variable that will receive the length of the serialized DNSKEY.

Returns:

a pointer to a newly allocated buffer holding the serialized DNSKEY on success, or `NULL` on failure.

Definition at line 2341 of file dns.c.

References `_encode_rsa_pubkey()`, `_mem_append()`, `_mem_append_serialized()`, `_mem_append_serialized_string()`, `dnskey::algorithm`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dnskey::is_sep`, `dnskey::is_zone`, `dnskey::keytag`, `dnskey::label`, `dnskey::pubkey`, `dnskey::rdata`, `dnskey::rdlen`, `RET_ERROR_PTR`, and `dnskey::validated`.

5.354.3.24 void* _serialize_ds_record_cb (void * *record*, size_t * *outlen*)

A callback record to serialize a DS record for storage in the object cache.

Definition at line 2233 of file dns.c.

References `_mem_append()`, `_mem_append_serialized()`, `_mem_append_serialized_string()`, `ds::algorithm`, `ds::digest`, `ds::digest_type`, `ds::diglen`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ds::keytag`, `ds::label`, `RET_ERROR_PTR`, and `ds::validated`.

5.354.3.25 int _sort_rrs_canonical (ns_msg * *dhandle*, int * *ordbuf*, size_t *nanswers*)

Sort the RRs in an RRSET into canonical order for DNSSEC signature verification.

Note:

This function does not actually sort the targeted RR's, but rather produces an output buffer containing an array of the sorted RR indices.

Parameters:

dhandle the handle of the DNS reply containing the answer RRs to be sorted.

ordbuf a pointer to an array of integers of size *nanswers* that will be updated to contain the new zero-indexed RR indices in proper canonically sorted order.

nanswers the number of answers in the DNS reply to be sorted (must match *dhandle*).

Returns:

-1 on error or 0 on success.

Definition at line 2518 of file dns.c.

References `_compare_rdata()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_RESOLVER`, and `RET_ERROR_INT`.

Referenced by `_rsa_verify_record()`.

- 5.354.3.26 PUBLIC_FUNC_DECL (void, free_mx_records, mx_record_t ** *mxs*)
- 5.354.3.27 PUBLIC_FUNC_DECL (mx_record_t **, get_mx_records, const char * *qstring*)
- 5.354.3.28 PUBLIC_FUNC_DECL (char *, get_txt_record, const char * *qstring*, unsigned long * *ttl*, int * *validated*)
- 5.354.3.29 PUBLIC_FUNC_DECL (void *, lookup_ds, const char * *label*)
- 5.354.3.30 PUBLIC_FUNC_DECL (void *, lookup_dnskey, const char * *label*)
- 5.354.3.31 PUBLIC_FUNC_DECL (int, validate_rrsig_rr, const char * *label*, ns_msg * *dhandle*, unsigned short *covered*, const unsigned char * *rdata*, size_t *rdlen*, dnskey_t ** *outkey*)
- 5.354.3.32 PUBLIC_FUNC_DECL (int, rsa_verify_record, const char * *label*, unsigned char *algorithm*, RSA * *pubkey*, const unsigned char * *rrsig*, const unsigned char * *sigbuf*, size_t *siglen*, ns_msg * *dhandle*)
- 5.354.3.33 PUBLIC_FUNC_DECL (void, destroy_ds, ds_t * *ds*)
- 5.354.3.34 PUBLIC_FUNC_DECL (void, destroy_dnskey, dnskey_t * *key*)
- 5.354.3.35 PUBLIC_FUNC_DECL (unsigned int, get_keytag, const unsigned char * *rdata*, size_t *rdlen*)
- 5.354.3.36 PUBLIC_FUNC_DECL (ds_t *, get_ds_by_dnskey, const dnskey_t * *key*)
- 5.354.3.37 PUBLIC_FUNC_DECL (dnskey_t *, get_dnskey_by_tag, unsigned int *tag*, const char * *signer*, int *force_lookup*)
- 5.354.3.38 PUBLIC_FUNC_DECL (int, compute_dnskey_sha_hash, const dnskey_t * *key*, size_t *nbits*, unsigned char * *outbuf*)
- 5.354.3.39 PUBLIC_FUNC_DECL (int, is_validated_key, dnskey_t * *dk*)
- 5.354.3.40 PUBLIC_FUNC_DECL (int, load_dnskey_file, const char * *filename*)

5.354.4 Variable Documentation

5.354.4.1 dnskey_rr_flags_t

Definition at line 68 of file dns.h.

Referenced by _add_dnskey_entry().

5.354.4.2 ds_rr_t

Definition at line 87 of file dns.h.

Referenced by _lookup_ds().

5.354.4.3 rrsig_rr_t

Definition at line 80 of file dns.h.

Referenced by _rsa_verify_record(), and _validate_rrsig_rr().

5.355 src/providers/dime/signet-resolver/mrec.c File Reference

```
#include "dime/signet-resolver/mrec.h"
#include "dime/signet-resolver/cache.h"
#include "dime/signet-resolver/dns.h"
#include "dime/common/misc.h"
#include "dime/common/error.h"
```

Functions

- `void _destroy_dime_record (dime_record_t *record)`
Destroy a DIME management record and its underlying data.
- `void _dump_dime_record_cb (FILE *fp, void *record, int brief)`
A callback handler to dump a DIME management record to the console.
- `void _destroy_dime_record_cb (void *record)`
A callback handler to destroy a DIME management record.
- `void * _deserialize_dime_record_cb (void *data, size_t len)`
A callback handler to deserialize a DIME record from the persistent cache for use in memory.
- `void * _serialize_dime_record_cb (void *record, size_t *outlen)`
A callback handler to serialize a DIME record for persistence through the data cache.
- `dime_record_t * _parse_dime_record (const char *txt, size_t len)`
Parse a data buffer into a DIME management record.
- `dime_record_t * _get_dime_record (const char *domain, unsigned long *ttl, int use_cache)`
Retrieve a DIME management record for a given dark domain via DNS.
- `int _validate_dime_record (const dime_record_t *record)`
Validate the syntax of a DIME record for correctness.
- `dime_record_t * _get_dime_record_from_file (const char *filename, const char *domain)`
Retrieve and parse a DIME management record from an input file.

5.355.1 Function Documentation

5.355.1.1 void* _deserialize_dime_record_cb (void * data, size_t len)

A callback handler to deserialize a DIME record from the persistent cache for use in memory.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

data a pointer to a serialized DIME record in binary form that will be deserialized.

len the length, in bytes, of the buffer to be deserialized.

Returns:

a pointer to a newly allocated DIME management record structure constructed from the serialized data, or NULL on failure.

Definition at line 196 of file mrec.c.

References `_deserialize_array()`, `_deserialize_data()`, `_deserialize_str_array()`, `_deserialize_string()`, `_destroy_dime_record()`, `_validate_dime_record()`, `dime_record_t::dx`, `ED25519_KEY_SIZE`, `ED25519_SIG_SIZE`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dime_record_t::expiry`, `dime_record_t::policy`, `dime_record_t::pubkey`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, `dime_record_t::subdomain`, `dime_record_t::syndicates`, `dime_record_t::tlssig`, `dime_record_t::validated`, and `dime_record_t::version`.

5.355.1.2 void _destroy_dime_record (dime_record_t * record)

Destroy a DIME management record and its underlying data.

Parameters:

record a pointer to the DIME management record object to be destroyed.

Definition at line 13 of file mrec.c.

References `_dbgprint()`, `_ptr_chain_free()`, `dime_record_t::dx`, `dime_record_t::pubkey`, `dime_record_t::syndicates`, and `dime_record_t::tlssig`.

Referenced by `_deserialize_dime_record_cb()`, `_destroy_dime_record_cb()`, `_get_dime_record()`, `_parse_dime_record()`, `_sgnt_resolv_destroy_dmtplib_session()`, and `_sgnt_resolv_dmtplib_connect()`.

5.355.1.3 void _destroy_dime_record_cb (void * record)

A callback handler to destroy a DIME management record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the DIME management record to be destroyed.

Definition at line 183 of file mrec.c.

References `_destroy_dime_record()`.

5.355.1.4 void _dump_dime_record_cb (FILE * fp, void * record, int brief)

A callback handler to dump a DIME management record to the console.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

fp a pointer to the file stream that will receive the dump output.

record a pointer to the DIME management record structure to be dumped.

brief if set, only print a brief one-line description for the requested record.

Definition at line 41 of file mrec.c.

References `_b64encode_nopad()`, `_clear_error_stack()`, `_hex_encode()`, `dump_error_stack()`, `dime_record_t::dx`, `ED25519_KEY_SIZE`, `ED25519_SIG_SIZE`, `dime_record_t::expiry`, `msg_experimental`, `msg_mixed`, `msg_strict`, `dime_record_t::policy`, `dime_record_t::pubkey`, `sub_explicit`, `sub_relaxed`, `sub_strict`, `dime_record_t::subdomain`, `dime_record_t::syndicates`, `dime_record_t::tlssig`, `dime_record_t::validated`, and `dime_record_t::version`.

5.355.1.5 `dime_record_t* _get_dime_record (const char * domain, unsigned long * ttl, int use_cache)`

Retrieve a DIME management record for a given dark domain via DNS.

Parameters:

domain a null-terminated string containing the name of the dark domain to be queried.

ttl an optional pointer to a variable that will receive the current TTL value of the DIME record's TXT RR.

use_cache if set, use the cache as the first-line resolver; if 0, only perform live network lookups.

Returns:

NULL on failure, or a pointer to a populated [dime_record_t](#) structure on success.

Definition at line 489 of file mrec.c.

References `_add_cached_object()`, `_clear_error_stack()`, `_create_cached_object()`, `_dbgprint()`, `_destroy_cache_entry()`, `_destroy_dime_record()`, `_find_cached_object()`, `_get_cache_obj_data()`, `_get_txt_record()`, `_is_object_expired()`, `_parse_dime_record()`, `_replace_object()`, `cached_data_drec`, `cached_stores`, `cached_object::data`, `DIME_RECORD_DNS_PREFIX`, `dump_error_stack()`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `cached_object::expiration`, `dime_record_t::expiry`, `get_last_error()`, `PUSH_ERROR`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, and `dime_record_t::validated`.

Referenced by `_sgnt_resolv_dmtplib_connect()`.

5.355.1.6 `dime_record_t* _get_dime_record_from_file (const char * filename, const char * domain)`

Retrieve and parse a DIME management record from an input file.

Parameters:

filename a null-terminated string containing the filename containing the DIME management record to be parsed.

domain a null-terminated string containing the name of the dark domain with which the DIME record is to be associated.

Returns:

NULL on failure or a pointer to the retrieved DIME management record on success.

Definition at line 631 of file mrec.c.

References `_add_cached_object_forced()`, `_clear_error_stack()`, `_get_cache_obj_data()`, `_parse_dime_record()`, `cached_data_drec`, `cached_stores`, `chr_isprint`, `dump_error_stack()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_SYSCALL`, and `RET_ERROR_PTR`.

5.355.1.7 `dime_record_t* _parse_dime_record (const char * txt, size_t len)`

Parse a data buffer into a DIME management record.

Note:

The data passed to this function is typically retrieved from DNS via the `_dx` TXT record.

Parameters:

txt a pointer to the data buffer to be parsed.

len the length, in bytes, of the data buffer to be parsed.

Returns:

a pointer to the DIME management record on success, or NULL on failure.

Definition at line 280 of file mrec.c.

References `_b64decode_nopad()`, `_dbgprint()`, `_destroy_dime_record()`, `_ptr_chain_add()`, `_validate_dime_record()`, `chr_isspace`, `DIME_VERSION_NO`, `dime_record_t::dx`, `ED25519_KEY_B64_SIZE`, `ED25519_KEY_SIZE`, `ED25519_SIG_B64_SIZE`, `ED25519_SIG_SIZE`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `msg_experimental`, `msg_mixed`, `msg_strict`, `dime_record_t::policy`, `dime_record_t::pubkey`, `PUSH_ERROR_FMT`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, `sub_explicit`, `sub_relaxed`, `sub_strict`, `dime_record_t::subdomain`, `dime_record_t::tlssig`, and `dime_record_t::version`.

Referenced by `_get_dime_record()`, and `_get_dime_record_from_file()`.

5.355.1.8 void* _serialize_dime_record_cb (void * record, size_t * outlen)

A callback handler to serialize a DIME record for persistence through the data cache.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to a DIME management record object to be serialized for persistence.

outlen a pointer to a variable to receive the length of the serialized data on completion.

Returns:

a pointer to a newly allocated buffer containing the serialized DIME record data, or NULL on failure.

Definition at line 241 of file mrec.c.

References `_mem_append()`, `_mem_append_serialized_array()`, `_mem_append_serialized_str_array()`, `_mem_append_serialized_string()`, `dime_record_t::dx`, `ED25519_KEY_SIZE`, `ED25519_SIG_SIZE`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `dime_record_t::expiry`, `dime_record_t::policy`, `dime_record_t::pubkey`, `RET_ERROR_PTR`, `dime_record_t::subdomain`, `dime_record_t::syndicates`, `dime_record_t::tlssig`, `dime_record_t::validated`, and `dime_record_t::version`.

5.355.1.9 int _validate_dime_record (const dime_record_t * record)

Validate the syntax of a DIME record for correctness.

Parameters:

record a pointer to the DIME management record to be validated.

Returns:

0 if the record was validated successfully or -1 on error.

Definition at line 595 of file mrec.c.

References `DIME_VERSION_NO`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `msg_experimental`, `msg_mixed`, `msg_strict`, `dime_record_t::policy`, `dime_record_t::pubkey`, `RET_ERROR_INT`, `RET_ERROR_INT_FMT`, `sub_explicit`, `sub_relaxed`, `sub_strict`, `dime_record_t::subdomain`, and `dime_record_t::version`.

Referenced by `_deserialize_dime_record_cb()`, and `_parse_dime_record()`.

5.356 src/providers/dime/signet-resolver/mrec.h File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include "dime/sds/sds.h"
#include <openssl/rsa.h>
#include "dime/common/dcrypto.h"
#include "dime/common/error.h"
```

Data Structures

- struct [dime_record_t](#)

Defines

- #define [DIME_VERSION_NO](#) 1
- #define [DIME_RECORD_DNS_PREFIX](#) "_dx"

Enumerations

- enum [dime_msg_policy](#) { [msg_experimental](#) = 0, [msg_mixed](#) = 1, [msg_strict](#) = 2 }
- enum [dime_sub_policy](#) { [sub_strict](#) = 0, [sub_relaxed](#) = 1, [sub_explicit](#) = 2 }

Functions

- [PUBLIC_FUNC_DECL](#) (void, destroy_dime_record, [dime_record_t](#) *record)
- [PUBLIC_FUNC_DECL](#) ([dime_record_t](#) *, parse_dime_record, const char *txt, size_t len)
- [PUBLIC_FUNC_DECL](#) ([dime_record_t](#) *, get_dime_record, const char *domain, unsigned long *ttl, int use_cache)
- [PUBLIC_FUNC_DECL](#) (int, validate_dime_record, const [dime_record_t](#) *record)
- [PUBLIC_FUNC_DECL](#) ([dime_record_t](#) *, get_dime_record_from_file, const char *filename, const char *domain)
- void [_destroy_dime_record_cb](#) (void *record)

A callback handler to destroy a DIME management record.

- void [_dump_dime_record_cb](#) (FILE *, void *record, int brief)

A callback handler to dump a DIME management record to the console.

- void * [_deserialize_dime_record_cb](#) (void *data, size_t len)

A callback handler to deserialize a DIME record from the persistent cache for use in memory.

- void * [_serialize_dime_record_cb](#) (void *record, size_t *outlen)

A callback handler to serialize a DIME record for persistence through the data cache.

5.356.1 Define Documentation

5.356.1.1 #define DIME_RECORD_DNS_PREFIX "_dx"

Definition at line 16 of file mrec.h.

Referenced by `_get_dime_record()`.

5.356.1.2 #define DIME_VERSION_NO 1

Definition at line 15 of file mrec.h.

Referenced by `_parse_dime_record()`, and `_validate_dime_record()`.

5.356.2 Enumeration Type Documentation

5.356.2.1 enum dime_msg_policy

Enumerator:

msg_experimental

msg_mixed

msg_strict

Definition at line 19 of file mrec.h.

5.356.2.2 enum dime_sub_policy

Enumerator:

sub_strict

sub_relaxed

sub_explicit

Definition at line 25 of file mrec.h.

5.356.3 Function Documentation

5.356.3.1 void* _deserialize_dime_record_cb (void * *data*, size_t *len*)

A callback handler to deserialize a DIME record from the persistent cache for use in memory.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

data a pointer to a serialized DIME record in binary form that will be deserialized.

len the length, in bytes, of the buffer to be deserialized.

Returns:

a pointer to a newly allocated DIME management record structure constructed from the serialized data, or NULL on failure.

Definition at line 196 of file mrec.c.

References `_deserialize_array()`, `_deserialize_data()`, `_deserialize_str_array()`, `_deserialize_string()`, `_destroy_dime_record()`, `_validate_dime_record()`, `dime_record_t::dx`, `ED25519_KEY_SIZE`, `ED25519_SIG_SIZE`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `ERR_UNSPEC`, `dime_record_t::expiry`, `dime_record_t::policy`, `dime_record_t::pubkey`, `PUSH_ERROR_SYSCALL`, `RET_ERROR_PTR`, `dime_record_t::subdomain`, `dime_record_t::syndicates`, `dime_record_t::tlssig`, `dime_record_t::validated`, and `dime_record_t::version`.

5.356.3.2 `void _destroy_dime_record_cb (void *record)`

A callback handler to destroy a DIME management record.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the DIME management record to be destroyed.

Definition at line 183 of file mrec.c.

References `_destroy_dime_record()`.

5.356.3.3 `void _dump_dime_record_cb (FILE *fp, void *record, int brief)`

A callback handler to dump a DIME management record to the console.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

fp a pointer to the file stream that will receive the dump output.

record a pointer to the DIME management record structure to be dumped.

brief if set, only print a brief one-line description for the requested record.

Definition at line 41 of file mrec.c.

References `_b64encode_nopad()`, `_clear_error_stack()`, `_hex_encode()`, `dump_error_stack()`, `dime_record_t::dx`, `ED25519_KEY_SIZE`, `ED25519_SIG_SIZE`, `dime_record_t::expiry`, `msg_experimental`, `msg_mixed`, `msg_strict`, `dime_record_t::policy`, `dime_record_t::pubkey`, `sub_explicit`, `sub_relaxed`, `sub_strict`, `dime_record_t::subdomain`, `dime_record_t::syndicates`, `dime_record_t::tlssig`, `dime_record_t::validated`, and `dime_record_t::version`.

5.356.3.4 `void* _serialize_dime_record_cb (void *record, size_t *outlen)`

A callback handler to serialize a DIME record for persistence through the data cache.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to a DIME management record object to be serialized for persistence.

outlen a pointer to a variable to receive the length of the serialized data on completion.

Returns:

a pointer to a newly allocated buffer containing the serialized DIME record data, or NULL on failure.

Definition at line 241 of file mrec.c.

References `_mem_append()`, `_mem_append_serialized_array()`, `_mem_append_serialized_str_array()`, `_mem_append_serialized_string()`, `dime_record_t::dx`, `ED25519_KEY_SIZE`, `ED25519_SIG_SIZE`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `dime_record_t::expiry`, `dime_record_t::policy`, `dime_record_t::pubkey`, `RET_ERROR_PTR`, `dime_record_t::subdomain`, `dime_record_t::syndicates`, `dime_record_t::tlssig`, `dime_record_t::validated`, and `dime_record_t::version`.

5.356.3.5 PUBLIC_FUNC_DECL (`dime_record_t *`, `get_dime_record_from_file`, `const char *filename`, `const char *domain`)

5.356.3.6 PUBLIC_FUNC_DECL (`int`, `validate_dime_record`, `const dime_record_t *record`)

5.356.3.7 PUBLIC_FUNC_DECL (`dime_record_t *`, `get_dime_record`, `const char *domain`, `unsigned long *ttl`, `int use_cache`)

5.356.3.8 PUBLIC_FUNC_DECL (`dime_record_t *`, `parse_dime_record`, `const char *txt`, `size_t len`)

5.356.3.9 PUBLIC_FUNC_DECL (`void`, `destroy_dime_record`, `dime_record_t *record`)

5.357 src/providers/dime/signet-resolver/mrec_pub.c File Reference

```
#include "dime/signet-resolver/mrec.h"
```

Functions

- void [destroy_dime_record](#) (dime_record_t *record)
- dime_record_t * [parse_dime_record](#) (const char *txt, size_t len)
- dime_record_t * [get_dime_record](#) (const char *domain, unsigned long *ttl, int use_cache)
- int [validate_dime_record](#) (const dime_record_t *record)
- dime_record_t * [get_dime_record_from_file](#) (const char *filename, const char *domain)

5.357.1 Function Documentation

5.357.1.1 void [destroy_dime_record](#) (dime_record_t * *record*)

Definition at line 3 of file mrec_pub.c.

References [PUBLIC_FUNC_IMPL_VOID](#).

5.357.1.2 dime_record_t* [get_dime_record](#) (const char * *domain*, unsigned long * *ttl*, int *use_cache*)

Definition at line 11 of file mrec_pub.c.

References [PUBLIC_FUNC_IMPL](#).

5.357.1.3 dime_record_t* [get_dime_record_from_file](#) (const char * *filename*, const char * *domain*)

Definition at line 19 of file mrec_pub.c.

References [PUBLIC_FUNC_IMPL](#).

5.357.1.4 dime_record_t* [parse_dime_record](#) (const char * *txt*, size_t *len*)

Definition at line 7 of file mrec_pub.c.

References [PUBLIC_FUNC_IMPL](#).

5.357.1.5 int [validate_dime_record](#) (const dime_record_t * *record*)

Definition at line 15 of file mrec_pub.c.

References [PUBLIC_FUNC_IMPL](#).

5.358 src/providers/dime/signet-resolver/signet-ssl.h File Reference

```
#include <unistd.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <openssl/rand.h>
#include <openssl/ocsp.h>
#include "dime/common/error.h"
```

Defines

- `#define CA_FILE "cacert.pem"`
- `#define CA_DIR "./"`
- `#define CRL_FILE "crl.pem"`

Functions

- `PUBLIC_FUNC_DECL` (int, ssl_initialize, void)
- `PUBLIC_FUNC_DECL` (void, ssl_shutdown, void)
- `PUBLIC_FUNC_DECL` (SSL *, ssl_connect_host, const char *hostname, unsigned short port, int force_family)
- `PUBLIC_FUNC_DECL` (void, ssl_disconnect, SSL *handle)
- `PUBLIC_FUNC_DECL` (SSL_CTX *, ssl_get_client_context, void)
- `PUBLIC_FUNC_DECL` (SSL *, ssl_starttls, int fd)
- `PUBLIC_FUNC_DECL` (int, do_x509_validation, X509 *cert, STACK_OF(X509)*chain)
- `PUBLIC_FUNC_DECL` (int, do_x509_hostname_check, X509 *cert, const char *domain)
- `PUBLIC_FUNC_DECL` (int, do_ocsp_validation, SSL *connection, int *fallthrough)
- `PUBLIC_FUNC_DECL` (char *, get_cert_subject_cn, X509 *cert)
- `void _ssl_fd_loop` (SSL *connection)

Create an fd loop (connect stdin to an ssl connection write fd, and ssl connection output to stdout).

- `int _verify_certificate_callback` (int ok, X509_STORE_CTX *ctx)

An internal callback function for DMTP TLS certificate verification.

- `int _validate_self_signed` (X509 *cert)

Validate a self-signed x509 certificate.

- `X509_STORE * _get_cert_store` (void)

Get an x509 certificate store populated with the root certificate bundle.

- `int _domain_wildcard_check` (const char *pattern, const char *domain)

Perform a check against a wildcard pattern for a domain name.

- `void _destroy_ocsp_response_cb` (void *record)

A callback handler to destroy an OCSP_RESPONSE object.

- `int _ocsp_response_callback` (SSL *s, void *arg)

An OCSP stapling rOCSP_REQUEST_freesponse callback for the TLS subsystem.

- `char * _get_cache_ocsp_id` (X509 *cert, OCSP_CERTID *cid, char *buf, size_t blen)

Get a unique identifier string for an OCSP response to be used in the object cache.

- void * [_serialize_ocsp_response_cb](#) (void *record, size_t *outlen)
A callback handler to serialize an OCSP response message.
- void * [_deserialize_ocsp_response_cb](#) (void *data, size_t len)
A callback handler to deserialize an OCSP response message.
- void [_dump_ocsp_response_cb](#) (FILE *fp, void *record, int brief)
A callback handler to dump an OCSP response to the console.

5.358.1 Define Documentation

5.358.1.1 `#define CA_DIR "/"`

Definition at line 15 of file signet-ssl.h.

5.358.1.2 `#define CA_FILE "cacert.pem"`

Definition at line 14 of file signet-ssl.h.

Referenced by `_ssl_initialize()`.

5.358.1.3 `#define CRL_FILE "crl.pem"`

Definition at line 16 of file signet-ssl.h.

Referenced by `_ssl_initialize()`.

5.358.2 Function Documentation

5.358.2.1 void* `_deserialize_ocsp_response_cb` (void * *data*, size_t *len*)

A callback handler to deserialize an OCSP response message.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

data a pointer to a buffer containing the data to be deserialized.

len the length, in bytes, of the data buffer to be deserialized.

Returns:

a pointer to a newly allocated OCSP_RESPONSE structure on success, or NULL on failure.

Definition at line 1478 of file ssl.c.

References `d2i_OCSP_RESPONSE_d`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, and `RET_ERROR_PTR`.

5.358.2.2 void _destroy_ocsp_response_cb (void * *record*)

A callback handler to destroy an OCSP_RESPONSE object.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the OCSP_RESPONSE object to be destroyed.

Definition at line 1274 of file ssl.c.

References OCSP_RESPONSE_free_d.

5.358.2.3 int _domain_wildcard_check (const char * *pattern*, const char * *domain*)

Perform a check against a wildcard pattern for a domain name.

Parameters:

pattern a pointer to the wildcard pattern string, subject as a certificate CN or SAN dnsname.

domain a pointer to the domain name, which will be compared against the supplied wildcard string.

Returns:

1 if the domain name matches the pattern or 0 if it does not; -1 if a general error was encountered.

Definition at line 1061 of file ssl.c.

References ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_INT.

5.358.2.4 void _dump_ocsp_response_cb (FILE * *fp*, void * *record*, int *brief*)

A callback handler to dump an OCSP response to the console.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

fp a pointer to the file stream that will receive the dump output.

record a pointer to the OCSP_RESPONSE object to be dumped.

brief if set, only print a brief one-line description for the requested record.

Definition at line 1344 of file ssl.c.

References ASN1_GENERALIZEDTIME_print_d, ASN1_INTEGER_to_BN_d, BIO_free_d, BIO_new_fp_d, BN_bn2hex_d, BN_free_d, ERR_print_errors_fp_d, OCSP_BASICRESP_free_d, OCSP_check_validity_d, OCSP_response_get1_basic_d, OCSP_response_status_d, OCSP_response_status_str_d, sk_num_d, sk_pop_d, and STACK_OF().

5.358.2.5 char* _get_cache_ocsp_id (X509 * *cert*, OCSP_CERTID * *cid*, char * *buf*, size_t *blen*)

Get a unique identifier string for an OCSP response to be used in the object cache.

Parameters:

- cert* a pointer to the X509 certificate undergoing verification.
- cid* a pointer to the OCSP certificate ID derived from the certificate and its issuer.
- buf* a pointer to a buffer that will hold the generated OCSP response ID in the cache.
- blen* the size, in bytes, of the buffer holding the OCSP response id.

Returns:

a pointer to the output buffer with the OCSP response ID on success, or NULL on failure.

Definition at line 1290 of file ssl.c.

References `_compute_sha_hash()`, `_get_cert_subject_cn()`, `CRYPTO_free_d`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `i2d_OCSP_CERTID_d`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, and `SHA_160_SIZE`.

Referenced by `_do_ocsp_validation()`.

5.358.2.6 X509_STORE* _get_cert_store (void)

Get an x509 certificate store populated with the root certificate bundle.

Returns:

NULL on failure, or a pointer to the x509 certificate store on success.

Definition at line 1124 of file ssl.c.

References `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, `store`, `X509_STORE_free_d`, `X509_STORE_load_locations_d`, and `X509_STORE_new_d`.

Referenced by `_do_ocsp_validation()`.

5.358.2.7 int _ocsp_response_callback (SSL * s, void * arg)

An OCSP stapling rOCSP_REQUEST_freesresponse callback for the TLS subsystem.

Note:

This function currently does not do anything!

Parameters:

- s* a pointer to the SSL connection that generated the callback.
- arg* an optional callback parameter that was set by the caller.

Returns:

1 if the OCSP response was valid or 0 if it was not.

Definition at line 1238 of file ssl.c.

References `_verbose`, `d2i_OCSP_RESPONSE_d`, `ERR_print_errors_fp_d`, `OCSP_RESPONSE_free_d`, `OCSP_RESPONSE_print_d`, and `SSL_ctrl_d`.

Referenced by `_ssl_connect_host()`.

5.358.2.8 void* _serialize_ocsp_response_cb (void * *record*, size_t * *outlen*)

A callback handler to serialize an OCSP response message.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the OCSP_RESPONSE object to be serialized.

outlen a pointer of a variable that will receive the length of the serialized OCSP response.

Returns:

a pointer to a newly allocated buffer holding the serialized OCSP_RESPONSE on success, or NULL on failure.

Definition at line 1502 of file ssl.c.

References ERR_BAD_PARAM, ERR_UNSPEC, i2d_OCSP_RESPONSE_d, PUSH_ERROR_OPENSSL, and RET_ERROR_PTR.

5.358.2.9 void _ssl_fd_loop (SSL * *connection*)

Create an fd loop (connect stdin to an ssl connection write fd, and ssl connection output to stdout).

Parameters:

connection the SSL connection handle to be passed to the control of the active console.

Definition at line 1147 of file ssl.c.

References ERR_print_errors_fp_d, SSL_get_rfd_d, SSL_read_d, and SSL_write_d.

5.358.2.10 int _validate_self_signed (X509 * *cert*)

Validate a self-signed x509 certificate.

Note:

This is completed by adding the certificate to a temporary cert store/context and calling internal validation.

Parameters:

cert a pointer to the x509 certificate to be validated.

Returns:

-1 on general failure, 0 on validation error, or 1 on success.

Definition at line 1005 of file ssl.c.

References _dbgprint(), ERR_UNSPEC, PUSH_ERROR_OPENSSL, RET_ERROR_INT, store, X509_STORE_CTX_free_d, X509_STORE_CTX_get_error_d, X509_STORE_CTX_init_d, X509_STORE_CTX_new_d, X509_STORE_free_d, X509_STORE_new_d, X509_verify_cert_d, and X509_verify_cert_error_string_d.

5.358.2.11 int _verify_certificate_callback (int *preverify_ok*, X509_STORE_CTX * *ctx*)

An internal callback function for DMTP TLS certificate verification.

Note:

This callback is set from [SSL_CTX_set_verify_d\(\)](#).

Parameters:

preverify_ok if set, indicates that the verification of the certificate passed, or 0 if it didn't.
ctx a pointer to the complete context used for the certificate chain verification.

Returns:

if 0, the verification process is stopped immediately, or if 1, the process is continued.

Definition at line 947 of file ssl.c.

References [_clear_error_stack\(\)](#), [_dbgprint\(\)](#), [_get_cert_subject_cn\(\)](#), [ERR_print_errors_fp_d](#), [X509_get_subject_name_d](#), [X509_NAME_online_d](#), [X509_STORE_CTX_get_current_cert_d](#), [X509_STORE_CTX_get_error_d](#), and [X509_STORE_CTX_get_error_depth_d](#).

Referenced by [_ssl_get_client_context\(\)](#).

5.358.2.12 PUBLIC_FUNC_DECL (char *, [get_cert_subject_cn](#), X509 * *cert*)

5.358.2.13 PUBLIC_FUNC_DECL (int, [do_ocsp_validation](#), SSL * *connection*, int * *fallthrough*)

5.358.2.14 PUBLIC_FUNC_DECL (int, [do_x509_hostname_check](#), X509 * *cert*, const char * *domain*)

5.358.2.15 PUBLIC_FUNC_DECL (int, [do_x509_validation](#), X509 * *cert*, STACK_OF(X509)* *chain*)

5.358.2.16 PUBLIC_FUNC_DECL (SSL *, [ssl_starttls](#), int *fd*)

5.358.2.17 PUBLIC_FUNC_DECL (SSL_CTX *, [ssl_get_client_context](#), void)

5.358.2.18 PUBLIC_FUNC_DECL (void, [ssl_disconnect](#), SSL * *handle*)

5.358.2.19 PUBLIC_FUNC_DECL (SSL *, [ssl_connect_host](#), const char * *hostname*, unsigned short *port*, int *force_family*)

5.358.2.20 PUBLIC_FUNC_DECL (void, [ssl_shutdown](#), void)

5.358.2.21 PUBLIC_FUNC_DECL (int, [ssl_initialize](#), void)

5.359 src/providers/dime/signet-resolver/ssl.c File Reference

```
#include "dime/common/network.h"
#include "dime/common/misc.h"
#include "dime/common/error.h"
#include "openssl/ocsp.h"
#include "openssl/x509.h"
#include "dime/signet-resolver/cache.h"
#include "dime/signet-resolver/dmtp.h"
#include "dime/signet-resolver/signet-ssl.h"
#include "providers/symbols.h"
```

Functions

- `int _ssl_initialize (void)`
Initialize all the dependent layers of the openssl library for use.
- `void _ssl_shutdown (void)`
Shutdown the openssl subsystem.
- `SSL_CTX * _ssl_get_client_context (void)`
Get an SSL context that will be used for all DMTP client connections.
- `SSL * _ssl_starttls (int fd)`
Negotiate a TLS session over an existing network socket connection.
- `SSL * _ssl_connect_host (const char *hostname, unsigned short port, int force_family)`
Connect to a specified host and port via a TLS-enabled service.
- `void _ssl_disconnect (SSL *handle)`
Terminate an SSL connection.
- `int _do_x509_validation (X509 *cert, STACK_OF(X509)*chain)`
Perform X509 validation on a certificate presented by a remote TLS service.
- `int _do_x509_hostname_check (X509 *cert, const char *domain)`
Check to see that an issued X509 certificate matches the expected target domain name.
- `int _do_ocsp_validation (SSL *connection, int *fallthrough)`
Perform OCSP validation on an x509 certificate.
- `char * _get_cert_subject_cn (X509 *cert)`
Get the subject common name (CN) attribute of an X509 certificate.
- `int _verify_certificate_callback (int preverify_ok, X509_STORE_CTX *ctx)`
An internal callback function for DMTP TLS certificate verification.
- `int _validate_self_signed (X509 *cert)`
Validate a self-signed x509 certificate.

- `int _domain_wildcard_check` (const char *pattern, const char *domain)
Perform a check against a wildcard pattern for a domain name.
- `X509_STORE * _get_cert_store` (void)
Get an x509 certificate store populated with the root certificate bundle.
- `void _ssl_fd_loop` (SSL *connection)
Create an fd loop (connect stdin to an ssl connection write fd, and ssl connection output to stdout).
- `int _ocsp_response_callback` (SSL *s, void *arg)
An OCSP stapling rOCSP_REQUEST_freeresponse callback for the TLS subsystem.
- `void _destroy_ocsp_response_cb` (void *record)
A callback handler to destroy an OCSP_RESPONSE object.
- `char * _get_cache_ocsp_id` (X509 *cert, OCSP_CERTID *cid, char *buf, size_t blen)
Get a unique identifier string for an OCSP response to be used in the object cache.
- `void _dump_ocsp_response_cb` (FILE *fp, void *record, int brief)
A callback handler to dump an OCSP response to the console.
- `void * _deserialize_ocsp_response_cb` (void *data, size_t len)
A callback handler to deserialize an OCSP response message.
- `void * _serialize_ocsp_response_cb` (void *record, size_t *outlen)
A callback handler to serialize an OCSP response message.

5.359.1 Function Documentation

5.359.1.1 void* _deserialize_ocsp_response_cb (void *data, size_t len)

A callback handler to deserialize an OCSP response message.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

data a pointer to a buffer containing the data to be deserialized.

len the length, in bytes, of the data buffer to be deserialized.

Returns:

a pointer to a newly allocated OCSP_RESPONSE structure on success, or NULL on failure.

Definition at line 1478 of file ssl.c.

References `d2i_OCSP_RESPONSE_d`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, and `RET_ERROR_PTR`.

5.359.1.2 void _destroy_ocsp_response_cb (void * *record*)

A callback handler to destroy an OCSP_RESPONSE object.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the OCSP_RESPONSE object to be destroyed.

Definition at line 1274 of file ssl.c.

References OCSP_RESPONSE_free_d.

5.359.1.3 int _do_ocsp_validation (SSL * *connection*, int * *fallthrough*)

Perform OCSP validation on an x509 certificate.

Parameters:

connection a pointer to the SSL connection associated with the peer certificate to be OCSP validated.

fallthrough an optional pointer to a variable that will be set if OCSP validation did not fail for various reasons (ie. OCSP was not available), but validation was not completed.

Returns:

-1 on general error, 0 if the certificate did not pass OCSP validation, or 1 if it did so successfully.

Definition at line 425 of file ssl.c.

References _add_cached_object(), _clear_error_stack(), _connect_host(), _dbgprint(), _find_cached_object(), _get_cache_ocsp_id(), _get_cert_store(), _save_cache_contents(), _unlink_object(), _verbose, ASN1_GENERALIZEDTIME_print_d, BIO_free_all_d, BIO_free_d, BIO_new_fp_d, BIO_new_socket_d, cached_data_ocsp, cached_stores, cached_object::data, dump_error_stack(), ERR_BAD_PARAM, ERR_NOMEM, ERR_print_errors_fp_d, ERR_UNSPEC, get_last_error(), OCSP_basic_verify_d, OCSP_BASICRESP_free_d, OCSP_cert_to_id_d, OCSP_check_nonce_d, OCSP_check_validity_d, OCSP_parse_url_d, OCSP_REQ_CTX_add1_header_d, OCSP_REQ_CTX_set1_req_d, OCSP_request_add0_id_d, OCSP_request_add1_nonce_d, OCSP_REQUEST_free_d, OCSP_REQUEST_new_d, OCSP_REQUEST_print_d, OCSP_resp_find_status_d, OCSP_RESPONSE_free_d, OCSP_response_get1_basic_d, OCSP_RESPONSE_print_d, OCSP_response_status_d, OCSP_response_status_str_d, OCSP_sendreq_nbio_d, OCSP_sendreq_new_d, PUSH_ERROR_OPENSSL, PUSH_ERROR_SYSCALL, RET_ERROR_INT, RET_ERROR_INT_FMT, sk_num_d, sk_value_d, SSL_get_peer_cert_chain_d, SSL_get_peer_certificate_d, STACK_OF(), status, store, X509_check_issued_d, X509_email_free_d, X509_get1_ocsp_d, and X509_STORE_free_d.

Referenced by _verify_dx_certificate().

5.359.1.4 int _do_x509_hostname_check (X509 * *cert*, const char * *domain*)

Check to see that an issued X509 certificate matches the expected target domain name.

See also:

[_domain_wildcard_check\(\)](#)

Note:

This function will check two parts of the specified certificate against the supplied domain name: 1. The CN attribute of the subject field.
2. The dnsName of the SAN extension

Parameters:

cert a pointer to the X509 certificate that will be matched against the specified domain name.

domain a null-terminated string containing the domain name to be matched against the target certificate.

Returns:

-1 on general error, 0 if a match did not occur, or 1 if the domain matched the certificate.

Definition at line 349 of file ssl.c.

References ERR_BAD_PARAM, RET_ERROR_INT, and X509_check_host_d.

Referenced by _verify_dx_certificate().

5.359.1.5 int _do_x509_validation (X509 * *cert*, STACK_OF(X509)* *chain*)

Perform X509 validation on a certificate presented by a remote TLS service.

Parameters:

cert a pointer to the x509 certificate to be chain-validated.

chain an optional pointer to the x509 certificate chain presented by the remote service; it will be added to the root certificate store already used for validation.

Returns:

-1 on general error, 0 if the certificate failed validation, or 1 if it passed validation successfully.

Definition at line 244 of file ssl.c.

References _dbgprint(), _verbose, ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_OPENSSL, RET_ERROR_INT, RET_ERROR_INT_FMT, store, X509_get_subject_name_d, X509_LOOKUP_file_d, X509_NAME_oneline_d, X509_STORE_add_lookup_d, X509_STORE_CTX_free_d, X509_STORE_CTX_init_d, X509_STORE_CTX_new_d, X509_STORE_CTX_set_chain_d, X509_STORE_free_d, X509_STORE_load_locations_d, X509_STORE_new_d, X509_STORE_set_flags_d, X509_verify_cert_d, and X509_verify_cert_error_string_d.

Referenced by _verify_dx_certificate().

5.359.1.6 int _domain_wildcard_check (const char * *pattern*, const char * *domain*)

Perform a check against a wildcard pattern for a domain name.

Parameters:

pattern a pointer to the wildcard pattern string, subject as a certificate CN or SAN dnsname.

domain a pointer to the domain name, which will be compared against the supplied wildcard string.

Returns:

1 if the domain name matches the pattern or 0 if it does not; -1 if a general error was encountered.

Definition at line 1061 of file ssl.c.

References ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_SYSCALL, and RET_ERROR_INT.

5.359.1.7 void _dump_ocsp_response_cb (FILE * *fp*, void * *record*, int *brief*)

A callback handler to dump an OCSP response to the console.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

- fp* a pointer to the file stream that will receive the dump output.
- record* a pointer to the OCSP_RESPONSE object to be dumped.
- brief* if set, only print a brief one-line description for the requested record.

Definition at line 1344 of file ssl.c.

References ASN1_GENERALIZEDTIME_print_d, ASN1_INTEGER_to_BN_d, BIO_free_d, BIO_new_fp_d, BN_bn2hex_d, BN_free_d, ERR_print_errors_fp_d, OCSP_BASICRESP_free_d, OCSP_check_validity_d, OCSP_response_get1_basic_d, OCSP_response_status_d, OCSP_response_status_str_d, sk_num_d, sk_pop_d, and STACK_OF().

5.359.1.8 char* _get_cache_ocsp_id (X509 * cert, OCSP_CERTID * cid, char * buf, size_t blen)

Get a unique identifier string for an OCSP response to be used in the object cache.

Parameters:

- cert* a pointer to the X509 certificate undergoing verification.
- cid* a pointer to the OCSP certificate ID derived from the certificate and its issuer.
- buf* a pointer to a buffer that will hold the generated OCSP response ID in the cache.
- blen* the size, in bytes, of the buffer holding the OCSP response id.

Returns:

a pointer to the output buffer with the OCSP response ID on success, or NULL on failure.

Definition at line 1290 of file ssl.c.

References _compute_sha_hash(), _get_cert_subject_cn(), CRYPTO_free_d, ERR_BAD_PARAM, ERR_UNSPEC, i2d_OCSP_CERTID_d, PUSH_ERROR_OPENSSL, RET_ERROR_PTR, and SHA_160_SIZE.

Referenced by _do_ocsp_validation().

5.359.1.9 X509_STORE* _get_cert_store (void)

Get an x509 certificate store populated with the root certificate bundle.

Returns:

NULL on failure, or a pointer to the x509 certificate store on success.

Definition at line 1124 of file ssl.c.

References ERR_UNSPEC, PUSH_ERROR_OPENSSL, RET_ERROR_PTR, store, X509_STORE_free_d, X509_STORE_load_locations_d, and X509_STORE_new_d.

Referenced by _do_ocsp_validation().

5.359.1.10 char* _get_cert_subject_cn (X509 * cert)

Get the subject common name (CN) attribute of an X509 certificate.

Parameters:

- cert* a pointer to the X509 certificate to have its subject field parsed.

Returns:

a pointer to the X509 certificate's CN as a null-terminated string on success, or NULL on failure.

Definition at line 896 of file ssl.c.

References ASN1_STRING_data_d, ERR_BAD_PARAM, ERR_UNSPEC, PUSH_ERROR_OPENSSL, PUSH_ERROR_SYSCALL, RET_ERROR_PTR, X509_get_subject_name_d, X509_NAME_ENTRY_get_data_d, X509_NAME_get_entry_d, and X509_NAME_get_index_by_NID_d.

Referenced by _get_cache_ocsp_id(), and _verify_certificate_callback().

5.359.1.11 int _ocsp_response_callback (SSL * *s*, void * *arg*)

An OCSP stapling rOCSP_REQUEST_freeesponse callback for the TLS subsystem.

Note:

This function currently does not do anything!

Parameters:

s a pointer to the SSL connection that generated the callback.

arg an optional callback parameter that was set by the caller.

Returns:

1 if the OCSP response was valid or 0 if it was not.

Definition at line 1238 of file ssl.c.

References _verbose, d2i_OCSP_RESPONSE_d, ERR_print_errors_fp_d, OCSP_RESPONSE_free_d, OCSP_RESPONSE_print_d, and SSL_ctrl_d.

Referenced by _ssl_connect_host().

5.359.1.12 void* _serialize_ocsp_response_cb (void * *record*, size_t * *outlen*)

A callback handler to serialize an OCSP response message.

Note:

This is an internal function used by the cache management subsystem.

Parameters:

record a pointer to the OCSP_RESPONSE object to be serialized.

outlen a pointer to a variable that will receive the length of the serialized OCSP response.

Returns:

a pointer to a newly allocated buffer holding the serialized OCSP_RESPONSE on success, or NULL on failure.

Definition at line 1502 of file ssl.c.

References ERR_BAD_PARAM, ERR_UNSPEC, i2d_OCSP_RESPONSE_d, PUSH_ERROR_OPENSSL, and RET_ERROR_PTR.

5.359.1.13 SSL* `_ssl_connect_host` (const char * *hostname*, unsigned short *port*, int *force_family*)

Connect to a specified host and port via a TLS-enabled service.

Parameters:

hostname the hostname of the server to which the TLS connection will be established.

port the numerical port number of the service to which the connection should be made.

force_family an optional address family (AF_INET or AF_INET6) to force the TCP connection to take.

Returns:

NULL on failure, or the SSL descriptor of the newly established TLS session on success.

Definition at line 159 of file ssl.c.

References `_connect_host()`, `_dbgprint()`, `_ocsp_response_callback()`, `_ssl_get_client_context()`, `ERR_BAD_PARAM`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, `SSL_connect_d`, `SSL_ctrl_d`, `SSL_CTX_callback_ctrl_d`, `SSL_CTX_ctrl_d`, `SSL_new_d`, and `SSL_set_fd_d`.

Referenced by `_dx_connect_standard()`.

5.359.1.14 void `_ssl_disconnect` (SSL * *handle*)

Terminate an SSL connection.

Parameters:

handle the SSL connection handle to be closed.

Definition at line 213 of file ssl.c.

References `_dbgprint()`, `ERR_print_errors_fp_d`, `SSL_free_d`, `SSL_get_fd_d`, and `SSL_shutdown_d`.

Referenced by `_dx_connect_standard()`, `_sgnt_resolv_destroy_dmtplib_session()`, `_sgnt_resolv_dmtplib_quit()`, and `_sgnt_resolv_read_dmtplib_line()`.

5.359.1.15 void `_ssl_fd_loop` (SSL * *connection*)

Create an fd loop (connect stdin to an ssl connection write fd, and ssl connection output to stdout).

Parameters:

connection the SSL connection handle to be passed to the control of the active console.

Definition at line 1147 of file ssl.c.

References `ERR_print_errors_fp_d`, `SSL_get_rfd_d`, `SSL_read_d`, and `SSL_write_d`.

5.359.1.16 SSL_CTX* `_ssl_get_client_context` (void)

Get an SSL context that will be used for all DMTP client connections.

Returns:

a pointer to the DMTP SSL client context on success, or NULL on failure.

Definition at line 74 of file ssl.c.

References `_dbgprint()`, `_ssl_initialize()`, `_verify_certificate_callback()`, `DMTP_V1_CIPHER_LIST`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, `SSL_CTX_ctrl_d`, `SSL_CTX_new_d`, `SSL_CTX_set_cipher_list_d`, `SSL_CTX_set_verify_d`, and `SSLv23_client_method_d`.

Referenced by `_ssl_connect_host()`, and `_ssl_starttls()`.

5.359.1.17 `int _ssl_initialize (void)`

Initialize all the dependent layers of the openssl library for use.

Returns:

0 on success or -1 on error.

Definition at line 27 of file `ssl.c`.

References `_dbgprint()`, `_get_dime_dir_location()`, `CA_FILE`, `CRL_FILE`, `ERR_UNSPEC`, `RET_ERROR_INT`, `SSL_library_init_d`, and `SSL_load_error_strings_d`.

Referenced by `_ssl_get_client_context()`.

5.359.1.18 `void _ssl_shutdown (void)`

Shutdown the openssl subsystem.

Definition at line 56 of file `ssl.c`.

References `_dbgprint()`, `ERR_free_strings_d`, and `SSL_CTX_free_d`.

5.359.1.19 `SSL* _ssl_starttls (int fd)`

Negotiate a TLS session over an existing network socket connection.

Note:

This function is called immediately after server confirmation of a STARTTLS command receipt.

Parameters:

fd the file descriptor of the network socket over which the TLS session will be initiated.

Returns:

NULL on failure, or the SSL descriptor of the newly established TLS session on success.

Definition at line 122 of file `ssl.c`.

References `_dbgprint()`, `_ssl_get_client_context()`, `ERR_UNSPEC`, `PUSH_ERROR_OPENSSL`, `RET_ERROR_PTR`, `SSL_connect_d`, `SSL_new_d`, and `SSL_set_fd_d`.

Referenced by `_sgnt_resolv_dmtpl_initiate_starttls()`.

5.359.1.20 `int _validate_self_signed (X509 * cert)`

Validate a self-signed x509 certificate.

Note:

This is completed by adding the certificate to a temporary cert store/context and calling internal validation.

Parameters:

cert a pointer to the x509 certificate to be validated.

Returns:

-1 on general failure, 0 on validation error, or 1 on success.

Definition at line 1005 of file ssl.c.

References [_dbgprint\(\)](#), [ERR_UNSPEC](#), [PUSH_ERROR_OPENSSL](#), [RET_ERROR_INT](#), [store](#), [X509_STORE_CTX_free_d](#), [X509_STORE_CTX_get_error_d](#), [X509_STORE_CTX_init_d](#), [X509_STORE_CTX_new_d](#), [X509_STORE_free_d](#), [X509_STORE_new_d](#), [X509_verify_cert_d](#), and [X509_verify_cert_error_string_d](#).

5.359.1.21 int _verify_certificate_callback (int *preverify_ok*, X509_STORE_CTX * *ctx*)

An internal callback function for DMTP TLS certificate verification.

Note:

This callback is set from [SSL_CTX_set_verify_d\(\)](#).

Parameters:

preverify_ok if set, indicates that the verification of the certificate passed, or 0 if it didn't.

ctx a pointer to the complete context used for the certificate chain verification.

Returns:

if 0, the verification process is stopped immediately, or if 1, the process is continued.

Definition at line 947 of file ssl.c.

References [_clear_error_stack\(\)](#), [_dbgprint\(\)](#), [_get_cert_subject_cn\(\)](#), [ERR_print_errors_fp_d](#), [X509_get_subject_name_d](#), [X509_NAME_online_d](#), [X509_STORE_CTX_get_current_cert_d](#), [X509_STORE_CTX_get_error_d](#), and [X509_STORE_CTX_get_error_depth_d](#).

Referenced by [_ssl_get_client_context\(\)](#).

5.360 src/providers/dime/signet-resolver/ssl_pub.c File Reference

```
#include "dime/signet-resolver/signet-ssl.h"
```

Functions

- int [ssl_initialize](#) (void)
- void [ssl_shutdown](#) (void)
- SSL_CTX * [ssl_get_client_context](#) (void)
- SSL * [ssl_starttls](#) (int fd)
- SSL * [ssl_connect_host](#) (const char *hostname, unsigned short port, int force_family)
- void [ssl_disconnect](#) (SSL *handle)
- int [do_x509_validation](#) (X509 *cert, STACK_OF(X509)*chain)
- int [do_x509_hostname_check](#) (X509 *cert, const char *domain)
- int [do_ocsp_validation](#) (SSL *connection, int *fallthrough)
- char * [get_cert_subject_cn](#) (X509 *cert)

5.360.1 Function Documentation

5.360.1.1 int do_ocsp_validation (SSL * *connection*, int * *fallthrough*)

Definition at line 36 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL.

5.360.1.2 int do_x509_hostname_check (X509 * *cert*, const char * *domain*)

Definition at line 32 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL.

5.360.1.3 int do_x509_validation (X509 * *cert*, STACK_OF(X509)* *chain*)

Definition at line 28 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL.

5.360.1.4 char* get_cert_subject_cn (X509 * *cert*)

Definition at line 40 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL.

5.360.1.5 SSL* ssl_connect_host (const char * *hostname*, unsigned short *port*, int *force_family*)

Definition at line 20 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL.

5.360.1.6 void ssl_disconnect (SSL * *handle*)

Definition at line 24 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.360.1.7 SSL_CTX* ssl_get_client_context (void)

Definition at line 12 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL.

5.360.1.8 int ssl_initialize (void)

Definition at line 4 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL.

5.360.1.9 void ssl_shutdown (void)

Definition at line 8 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL_VOID.

5.360.1.10 SSL* ssl_starttls (int *fd*)

Definition at line 16 of file ssl_pub.c.

References PUBLIC_FUNC_IMPL.

5.361 src/providers/dime/signet/general.c File Reference

```
#include "dime/signet/common.h"
```

Defines

- #define [SKEY_EMPTY](#) { 0, 0, 0, 0, 0, 0, NULL, NULL }
- #define [SKEY_SIZE1](#) { 0, 1, 0, 1, 0, UNICODE, NULL, NULL }
- #define [SKEY_SIZE2](#) { 0, 1, 0, 2, 0, UNICODE, NULL, NULL }

Functions

- const char * [dime_number_to_str](#) (dime_number_t number)

Returns a string from a dime_number_t enum type.

Variables

- [signet_field_key_t](#) [signet_org_field_keys](#) [256]
- [signet_field_key_t](#) [signet_user_field_keys](#) [256]
- [signet_field_key_t](#) [signet_ssr_field_keys](#) [256]

5.361.1 Define Documentation

5.361.1.1 #define [SKEY_EMPTY](#) { 0, 0, 0, 0, 0, 0, NULL, NULL }

Definition at line 4 of file general.c.

5.361.1.2 #define [SKEY_SIZE1](#) { 0, 1, 0, 1, 0, UNICODE, NULL, NULL }

Definition at line 5 of file general.c.

5.361.1.3 #define [SKEY_SIZE2](#) { 0, 1, 0, 2, 0, UNICODE, NULL, NULL }

Definition at line 6 of file general.c.

5.361.2 Function Documentation

5.361.2.1 const char* [dime_number_to_str](#) (dime_number_t *number*)

Returns a string from a dime_number_t enum type.

Parameters:

number Dime number input.

Returns:

Null terminated string corresponding to the dime number.

Definition at line 1070 of file general.c.

References [DIME_ENCRYPTED_MSG](#), [DIME_ENCRYPTED_ORG_KEYS](#), [DIME_ENCRYPTED_USER_KEYS](#), [DIME_MSG_TRACING](#), [DIME_ORG_KEYS](#), [DIME_ORG_SIGNET](#), [DIME_SSR](#), [DIME_USER_KEYS](#), and [DIME_USER_SIGNET](#).

5.361.3 Variable Documentation

5.361.3.1 `signet_field_key_t signet_org_field_keys[256]`

Definition at line 18 of file general.c.

5.361.3.2 `signet_field_key_t signet_ssr_field_keys[256]`

Definition at line 785 of file general.c.

5.361.3.3 `signet_field_key_t signet_user_field_keys[256]`

Definition at line 405 of file general.c.

5.362 src/providers/dime/signet/signet.c File Reference

```
#include "string.h"
#include "dime/common/misc.h"
#include "dime/signet/keys.h"
#include "dime/signet/signet.h"
```

Data Structures

- struct [signet_field_t](#)

Functions

- EC_KEY * [dime_sgnt_enckey_fetch](#) ([signet_t](#) const *signet)
Retrieves the public encryption key from the signet, if the signet is a user signet only retrieves the main encryption key (not alternate).
- int [dime_sgnt_enckey_set](#) ([signet_t](#) *signet, EC_KEY *key, unsigned char format)
Sets the public encryption key (non-alterante encryption key) for the signet.
- int [dime_sgnt_fid_count_get](#) ([signet_t](#) const *signet, unsigned char fid)
Retrieves the number of fields with the specified field id.
- int [dime_sgnt_fid_exists](#) ([signet_t](#) const *signet, unsigned char fid)
Checks for presence of field with specified id in the signet.
- unsigned char * [dime_sgnt_fid_num_fetch](#) ([signet_t](#) const *signet, unsigned char fid, unsigned int num, size_t *data_size)
Fetches the binary data value of the field specified by field id and the number at which it appears in the signet amongst fields with the same field id (1, 2, ...).
- int [dime_sgnt_fid_num_remove](#) ([signet_t](#) *signet, unsigned char fid, int num)
Removes the field specified by a field id and the number in which it appears in the target signet amongst fields with the same field id from the target signet.
- int [dime_sgnt_field_defined_create](#) ([signet_t](#) *signet, unsigned char fid, size_t data_size, unsigned char const *data)
Adds a field to the target field.
- int [dime_sgnt_field_defined_set](#) ([signet_t](#) *signet, unsigned char fid, size_t data_size, const unsigned char *data)
Replaces all fields in the target signet with the specified field id with a new field specified by the parameters.
- int [dime_sgnt_field_undefined_create](#) ([signet_t](#) *signet, size_t name_size, unsigned char const *name, size_t data_size, unsigned char const *data)
Adds an undefined field to signet with specified name and data.
- unsigned char * [dime_sgnt_field_undefined_fetch](#) ([signet_t](#) const *signet, size_t name_size, unsigned char const *name, size_t *data_size)
Fetches the first undefined field with the specified field name.
- int [dime_sgnt_field_undefined_remove](#) ([signet_t](#) *signet, size_t name_size, unsigned char const *name)
Removes an undefined field from the target signet by name.
- int [dime_sgnt_file_create](#) ([signet_t](#) *signet, char const *filename)

Stores a signet from the [signet_t](#) structure in a PEM formatted file specified by the filename.

- char * [dime_sgnt_fingerprint_crypto](#) ([signet_t](#) const *signet)
Takes a SHA512 fingerprint of a signet with all fields after the cryptographic signature field stripped off.
- char * [dime_sgnt_fingerprint_full](#) ([signet_t](#) const *signet)
Takes a SHA512 fingerprint of the user or org signet with the ID and FULL signature fields stripped off.
- char * [dime_sgnt_fingerprint_id](#) ([signet_t](#) const *signet)
Takes a SHA512 fingerprint of the entire user or org signet.
- char * [dime_sgnt_fingerprint_ssr](#) ([signet_t](#) const *signet)
Takes a SHA512 fingerprint of a user signet or an ssr with all fields after the SSR signature stripped off.
- char * [dime_sgnt_id_fetch](#) ([signet_t](#) *signet)
Retrieves signet id.
- int [dime_sgnt_id_set](#) ([signet_t](#) *signet, size_t id_size, unsigned char const *id)
Sets the ID of the signet to the specified NULL terminated string.
- int [dime_sgnt_msg_sig_verify](#) ([signet_t](#) const *signet, [ed25519_signature](#) sig, unsigned char const *buf, size_t buf_len)
Uses a signet's signing keys to verify a signature.
- int [dime_sgnt_sig_coc_sign](#) ([signet_t](#) *signet, [ED25519_KEY](#) *key)
Checks for the presence of all required fields that come before the chain of custody signature field and adds the SSR signature.
- int [dime_sgnt_sig_crypto_sign](#) ([signet_t](#) *signet, [ED25519_KEY](#) *key)
Signs an SSR or an incomplete ORG signet with the cryptographic signature after checking for the presence of all previous required fields.
- int [dime_sgnt_sig_full_sign](#) ([signet_t](#) *signet, [ED25519_KEY](#) *key)
Checks for the presence of all required fields that come before the full signature and signs all the fields that come before the CORE signature field.
- int [dime_sgnt_sig_id_sign](#) ([signet_t](#) *signet, [ED25519_KEY](#) *key)
Checks for the presence of all required fields that come before the FULL signature and signs the entire target signet using the specified key.
- int [dime_sgnt_sig_ssr_sign](#) ([signet_t](#) *signet, [ED25519_KEY](#) *key)
Checks for the presence of all required fields that come before the SSR signature field and adds the SSR signature.
- [signet_t](#) * [dime_sgnt_signet_binary_deserialize](#) (unsigned char const *in, size_t len)
Returns a new [signet_t](#) structure that gets deserialized from a data buffer.
- unsigned char * [dime_sgnt_signet_binary_serialize](#) ([signet_t](#) *signet, uint32_t *serial_size)
Serializes a signet structure into binary data.
- [signet_t](#) * [dime_sgnt_signet_b64_deserialize](#) (const char *b64_in)
Deserializes a b64 signet into a signet structure.
- char * [dime_sgnt_signet_b64_serialize](#) ([signet_t](#) *signet)
Serializes a signet structure into b64 data.
- [signet_t](#) * [dime_sgnt_signet_create](#) ([signet_type_t](#) type)
Returns a new [signet_t](#) structure.

- [signet_t * dime_sgnt_signet_create_w_keys](#) ([signet_type_t](#) type, char const *keyfile)
Creates a signet structure with public signing and encryption keys. Also creates a keys file in which the private keys are stored.
- [signet_t * dime_sgnt_signet_crypto_split](#) ([signet_t](#) const *signet)
Creates a copy of the target user signet with all fields beyond the INITIAL signature stripped off.
- void [dime_sgnt_signet_destroy](#) ([signet_t](#) *signet)
Destroys a [signet_t](#) structure.
- void [dime_sgnt_signet_dump](#) (FILE *fp, [signet_t](#) *signet)
Dumps signet into the specified file descriptor.
- [signet_t * dime_sgnt_signet_dupe](#) ([signet_t](#) *signet)
Create a copy of the provided signet.
- [signet_t * dime_sgnt_signet_full_split](#) ([signet_t](#) const *signet)
Creates a copy of the target signet with the ID field and the FULL signature stripped off.
- [signet_t * dime_sgnt_signet_load](#) (char const *filename)
Loads [signet_t](#) structure from a PEM formatted file specified by filename.
- [ED25519_KEY * dime_sgnt_signkey_fetch](#) ([signet_t](#) const *signet)
Retrieves the public signing key from the signet, if the signet is an org signet only retrieves the POK.
- int [dime_sgnt_signkey_set](#) ([signet_t](#) *signet, [ED25519_KEY](#) *key, unsigned char format)
Retrieves all the signing keys from an org signet that can be used to sign a message.
- [ED25519_KEY ** dime_sgnt_signkeys_msg_fetch](#) ([signet_t](#) const *signet)
Retrieves all the signing keys from an org signet that can be used to sign a signet.
- [ED25519_KEY ** dime_sgnt_signkeys_signet_fetch](#) ([signet_t](#) const *signet)
Retrieves all the signing keys from an org signet that can be used to sign software.
- [ED25519_KEY ** dime_sgnt_signkeys_software_fetch](#) ([signet_t](#) const *signet)
Retrieves all the signing keys from an org signet that can be used to sign a TLS certificate.
- [ED25519_KEY ** dime_sgnt_signkeys_tls_fetch](#) ([signet_t](#) const *signet)
sets the signing key (pok - primary signing key in case of an org signet).
- int [dime_sgnt_sok_create](#) ([signet_t](#) *signet, [ED25519_KEY](#) *key, unsigned char format, uint8_t perm)
adds a sok (secondary organizational signing key) to an organizational signet.
- [ED25519_KEY * dime_sgnt_sok_num_fetch](#) ([signet_t](#) const *signet, unsigned int num)
fetch the secondary organizational signing key from the signet by number (starting at 1)
- char const * [dime_sgnt_state_to_str](#) ([signet_state_t](#) state)
returns a string from a [signet_state_t](#) enum type.
- [signet_type_t dime_sgnt_type_get](#) (const [signet_t](#) *signet)
retrieves the signet type, org or user ([signet_type_org](#) or [signet_type_user](#))
- int [dime_sgnt_type_set](#) ([signet_t](#) *signet, [signet_type_t](#) type)
sets the target signet to a specified type.

- `signet_state_t dime_sgnt_validate_all` (`signet_t` const *`signet`, `signet_t` const *`previous`, `signet_t` const *`orgsig`, unsigned char const **`dime_pok`)

verifies a user signet, org signet or ssr for syntax, context and cryptographic validity. does not perform chain of custody validation.

5.362.1 Function Documentation

5.362.1.1 EC_KEY* dime_sgnt_enckey_fetch (signet_t const * *signet*)

Retrieves the public encryption key from the signet, if the signet is a user signet only retrieves the main encryption key (not alternate).

Parameters:

signet Pointer to the target signet.

Returns:

Pointer to the target encryption public key. {free_ec_key}

Definition at line 3909 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.2 int dime_sgnt_enckey_set (signet_t * *signet*, EC_KEY * *key*, unsigned char *format*)

Sets the public encryption key (non-alterante encryption key) for the signet.

Parameters:

signet Target signet.

key Public encryption key.

format Format specifier. TODO currently unused! (spec requires 0x04 but openssl natively serializes it to 0x02).

Returns:

0 on success, -1 on failure.

Definition at line 3926 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.3 int dime_sgnt_fid_count_get (signet_t const * *signet*, unsigned char *fid*)

Retrieves the number of fields with the specified field id.

Parameters:

signet Pointer to the target signet.

fid The target field id.

Returns:

The number of fields with specified field id. On various errors returns -1. NOTE: int overflow should not occur because of field size lower and signet size upper bounds.

Definition at line 3942 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.4 `int dime_sgnt_fid_exists (signet_t const * signet, unsigned char fid)`

Checks for presence of field with specified id in the signet.

Parameters:

signet The signet to be checked

fid Specified field id

Returns:

1 if such a field exists, 0 if it does not exist, -1 if error.

Definition at line 3956 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.5 `unsigned char* dime_sgnt_fid_num_fetch (signet_t const * signet, unsigned char fid, unsigned int num, size_t * data_size)`

Fetches the binary data value of the field specified by field id and the number at which it appears in the signet amongst fields with the same field id (1, 2, ...).

Parameters:

signet Pointer to the target signet.

fid Specified field id.

num Specified field number based on the order in which it appears in the signet.

data_size Pointer to the length of returned array.

Returns:

Array containing the binary data of the specified field, NULL if an error occurs. {free}

Definition at line 3978 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.6 `int dime_sgnt_fid_num_remove (signet_t * signet, unsigned char fid, int num)`

Removes the field specified by a field id and the number in which it appears in the target signet amongst fields with the same field id from the target signet.

Parameters:

signet Pointer to the target signet.

fid Field id of the field to be removed.

num The number in which the field to be removed appears amongst other fields with the same field id in the target signet, (1, 2, ...).

Returns:

0 on success, -1 on failure.

Definition at line 3996 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.7 int dime_sgnt_field_defined_create (signet_t * *signet*, unsigned char *fid*, size_t *data_size*, unsigned char const * *data*)

Adds a field to the target field.

Parameters:

signet Pointer to the target signet.
fid Field id of the field to be added.
data_size Size of the array containing the field data.
data Field data.

Returns:

0 on success, -1 on failure.

Definition at line 4012 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.8 int dime_sgnt_field_defined_set (signet_t * *signet*, unsigned char *fid*, size_t *data_size*, const unsigned char * *data*)

Replaces all fields in the target signet with the specified field id with a new field specified by the parameters.

Parameters:

signet Pointer to the target [signet_t](#) structure.
fid Field id which specifies the fields to be replaced with the new field.
data_size Size of field data array.
data Array containing field data.

Returns:

0 on success, -1 on failure.

Definition at line 4030 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.9 int dime_sgnt_field_undefined_create (signet_t * *signet*, size_t *name_size*, unsigned char const * *name*, size_t *data_size*, unsigned char const * *data*)

Adds an undefined field to signet with specified name and data.

Parameters:

signet Pointer to the target signet to which the field is added.
name_size Size of field name.
name Pointer to field name.
data_size Size of field data.
data Pointer to field data.

Returns:

0 on success, -1 on failure.

Definition at line 4050 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.10 unsigned char* dime_sgnt_field_undefined_fetch (signet_t const * *signet*, size_t *name_size*, unsigned char const * *name*, size_t * *data_size*)

Fetches the first undefined field with the specified field name.

Parameters:

signet Pointer to the target signet.

name_size Length of the passed array containing the length of the target field name.

name Array containing the name of the desired undefined field.

data_size Pointer to the size of the array that gets returned by the function.

Returns:

The array containing the data from the specified field or NULL in case of failure such as if the field was not found. {free}

Definition at line 4069 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.11 int dime_sgnt_field_undefined_remove (signet_t * *signet*, size_t *name_size*, unsigned char const * *name*)

Removes an undefined field from the target signet by name.

Parameters:

signet Pointer to the target signet.

name_size Size of field name.

name Name of the field to be removed.

Returns:

0 on success, -1 on failure.

Definition at line 4085 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.12 int dime_sgnt_file_create (signet_t * *signet*, char const * *filename*)

Stores a signet from the [signet_t](#) structure in a PEM formatted file specified by the filename.

Parameters:

signet Pointer to the [signet_t](#) structure containing the signet.

filename NULL terminated string containing the desired filename for the signet.

Returns:

0 on success, -1 on failure.

Definition at line 4100 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.13 char* dime_sgnt_fingerprint_crypto (signet_t const * *signet*)

Takes a SHA512 fingerprint of a signet with all fields after the cryptographic signature field stripped off.

Note:

To take an SSR fingerprint, use the `signet_ssr_fingerprint()` function.

Parameters:

signet Pointer to the target signet.

Returns:

Allocated NULL terminated string to a base64 encoded unpadded fingerprint. NULL on error. {free}

Definition at line 4118 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.362.1.14 char* dime_sgnt_fingerprint_full (signet_t const * *signet*)

Takes a SHA512 fingerprint of the user or org signet with the ID and FULL signature fields stripped off.

Note:

To take an SSR fingerprint, use the `signet_ssr_fingerprint()` function.

Parameters:

signet Pointer to the target signet.

Returns:

Allocated NULL terminated buffer to a base64 encoded unpadded fingerprint. NULL on failure. {free}

Definition at line 4136 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.362.1.15 char* dime_sgnt_fingerprint_id (signet_t const * *signet*)

Takes a SHA512 fingerprint of the entire user or org signet.

Parameters:

signet Pointer to the target signet.

Returns:

Allocated NULL terminated buffer to a base64 encoded unpadded fingerprint. NULL on failure; {free}

Definition at line 4151 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.362.1.16 char* dime_sgnt_fingerprint_ssr (signet_t const * *signet*)

Takes a SHA512 fingerprint of a user signet or an ssr with all fields after the SSR signature stripped off.

Parameters:

signet Pointer to the target signet.

Returns:

Allocated NULL terminated buffer to a base64 encoded unpadded fingerprint. {free}

Definition at line 4166 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.17 char* dime_sgnt_id_fetch (signet_t * *signet*)

Retrieves signet id.

Parameters:

signet Signet from which the id is retrieved.

Returns:

Cstring containing ID or NULL on error. {free}

Definition at line 4180 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.18 int dime_sgnt_id_set (signet_t * *signet*, size_t *id_size*, unsigned char const * *id*)

Sets the ID of the signet to the specified NULL terminated string.

Parameters:

signet Pointer to the target signet.

id_size Size of signet id.

id Signet id.

Returns:

0 on success, -1 on failure.

Definition at line 4196 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.19 int dime_sgnt_msg_sig_verify (signet_t const * *signet*, ed25519_signature *sig*, unsigned char const * *buf*, size_t *buf_len*)

Uses a signet's signing keys to verify a signature.

Parameters:

signet Pointer to the signet.

sig ed25519 signature buffer to be verified.

buf Data buffer over which the signature was taken.

buf_len Length of data buffer.

Returns:

1 on successful verification, 0 if the signature could not be verified, -1 if an error occurred.

Definition at line 4215 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.20 int dime_sgnt_sig_coc_sign (signet_t * *signet*, ED25519_KEY * *key*)

Checks for the presence of all required fields that come before the chain of custody signature field and adds the SSR signature.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4230 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.21 int dime_sgnt_sig_crypto_sign (signet_t * *signet*, ED25519_KEY * *key*)

Signs an SSR or an incomplete ORG signet with the cryptographic signature after checking for the presence of all previous required fields.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4245 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.22 int dime_sgnt_sig_full_sign (signet_t * *signet*, ED25519_KEY * *key*)

Checks for the presence of all required fields that come before the full signature and signs all the fields that come before the CORE signature field.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4261 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.23 int dime_sgnt_sig_id_sign (signet_t * *signet*, ED25519_KEY * *key*)

Checks for the presence of all required fields that come before the FULL signature and signs the entire target signet using the specified key.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4275 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.24 int dime_sgnt_sig_ssr_sign (signet_t * *signet*, ED25519_KEY * *key*)

Checks for the presence of all required fields that come before the SSR signature field and adds the SSR signature.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4290 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.25 signet_t* dime_sgnt_signet_b64_deserialize (const char * *b64_in*)

Deserializes a b64 signet into a signet structure.

Parameters:

b64_in NULL terminated array of b64 signet data.

Returns:

Pointer to newly allocated signet structure, NULL if failure. {dime_sgnt_destroy_signet}

Definition at line 4336 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

Referenced by _get_signet().

5.362.1.26 char* dime_sgnt_signet_b64_serialize (signet_t * *signet*)

Serializes a signet structure into b64 data.

Parameters:

signet Pointer to the target signet.

Returns:

Signet serialized into b64 data. NULL on error. {free}

Definition at line 4350 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.27 `signet_t* dime_sgnt_signet_binary_deserialize (unsigned char const * in, size_t len)`

Returns a new [signet_t](#) structure that gets deserialized from a data buffer.

Parameters:

in data buffer containing the binary form of a signet

in_len length of data buffer

Returns:

A pointer to a newly allocated [signet_t](#) structure type, NULL on failure. {dime_sgnt_destroy_signet}

Definition at line 4306 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

Referenced by _deserialize_signet_cb().

5.362.1.28 `unsigned char* dime_sgnt_signet_binary_serialize (signet_t * signet, uint32_t * serial_size)`

Serializes a signet structure into binary data.

Parameters:

signet Pointer to the target signet.

serial_size Pointer to the value that stores the length of the array returned.

Returns:

Signet serialized into binary data. NULL on error. {free}

Definition at line 4322 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

Referenced by _serialize_signet_cb().

5.362.1.29 `signet_t* dime_sgnt_signet_create (signet_type_t type)`

Returns a new [signet_t](#) structure.

Parameters:

type signet type user org or sss (SIGNET_TYPE_USER, SIGNET_TYPE_ORG or SIGNET_TYPE_SSR)

Returns:

A pointer to a newly allocated [signet_t](#) structure type, NULL if failure. {dime_sgnt_destroy_signet}

Definition at line 4364 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.30 `signet_t* dime_sgnt_signet_create_w_keys (signet_type_t type, char const * keysfile)`

Creates a signet structure with public signing and encryption keys. Also creates a keys file in which the private keys are stored.

Parameters:

type Signet type, org, user or ssr (SIGNET_TYPE_ORG, SIGNET_TYPE_USER or SIGNET_TYPE_SSR).

keysfile NULL terminated string containing the name of the keyfile to be created.

Returns:

Pointer to the newly created and allocated [signet_t](#) structure or NULL on error. {dime_sgnt_destroy_signet}

Definition at line 4383 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.31 `signet_t* dime_sgnt_signet_crypto_split (signet_t const * signet)`

Creates a copy of the target user signet with all fields beyond the INITIAL signature stripped off.

Parameters:

signet Pointer to the target signet.

Returns:

Pointer to a stripped signet on success, NULL on failure. {dime_sgnt_destroy_signet}

Definition at line 4398 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.32 `void dime_sgnt_signet_destroy (signet_t * signet)`

Destroys a [signet_t](#) structure.

Parameters:

signet Pointer to the signet to be destroyed.

Definition at line 4408 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

Referenced by `_destroy_signet_cb()`, and `_get_signet()`.

5.362.1.33 `void dime_sgnt_signet_dump (FILE * fp, signet_t * signet)`

Dumps signet into the specified file descriptor.

Parameters:

fp File descriptor the signet is dumped to.

signet Pointer to the [signet_t](#) structure to be dumped.

Definition at line 4420 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

Referenced by `_dump_signet_cb()`.

5.362.1.34 `signet_t* dime_sgnt_signet_dupe (signet_t * signet)`

Create a copy of the provided signet.

Parameters:

signet Signet to be copied.

Returns:

The copy. {dime_sgnt_destroy_signet}

Definition at line 4434 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.35 `signet_t* dime_sgnt_signet_full_split (signet_t const * signet)`

Creates a copy of the target signet with the ID field and the FULL signature stripped off.

Parameters:

signet Pointer to the target signet.

Returns:

Pointer to a stripped signet on success, NULL on failure. {dime_sgnt_destroy_signet}

Definition at line 4449 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.36 `signet_t* dime_sgnt_signet_load (char const * filename)`

Loads [signet_t](#) structure from a PEM formatted file specified by filename.

Parameters:

filename NULL terminated string containing the filename of the file containing the signet.

Returns:

Pointer to a newly created [signet_t](#) structure loaded from the file, NULL on failure. {dime_sgnt_destroy_signet}

Definition at line 4465 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.37 `ED25519_KEY* dime_sgnt_signkey_fetch (signet_t const * signet)`

Retrieves the public signing key from the signet, if the signet is an org signet only retrieves the POK.

Parameters:

signet Pointer to the target signet.

Returns:

Pointer to the target ed25519 public key. {free_ed25519_key}

Definition at line 4480 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.38 int dime_sgmt_signkey_set (signet_t * *signet*, ED25519_KEY * *key*, unsigned char *format*)

Retrieves all the signing keys from an org signet that can be used to sign a message.

Parameters:

signet Pointer to target organizational signet.

Returns:

A NULL pointer terminated array of ed25519 public signing key objects.

Note:

Always returns at least POK. {free_ed25519_key_chain}

Definition at line 4496 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.39 ED25519_KEY dime_sgmt_signkeys_msg_fetch (signet_t const * *signet*)**

Retrieves all the signing keys from an org signet that can be used to sign a signet.

Parameters:

signet Pointer to target organizational signet.

Returns:

A NULL pointer terminated array of ed25519 public signing key objects.

Note:

Always returns at least POK. {free_ed25519_key_chain}

Definition at line 4513 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.40 ED25519_KEY dime_sgmt_signkeys_signet_fetch (signet_t const * *signet*)**

Retrieves all the signing keys from an org signet that can be used to sign software.

Parameters:

signet Pointer to target organizational signet.

Returns:

A NULL pointer terminated array of ed25519 public signing key objects.

Note:

Always returns at least POK. {free_ed25519_key_chain}

Definition at line 4530 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.41 ED25519_KEY dime_sgnt_signkeys_software_fetch (signet_t const * *signet*)**

Retrieves all the signing keys from an org signet that can be used to sign a TLS certificate.

Parameters:

signet Pointer to target organizational signet.

Returns:

A NULL pointer terminated array of ed25519 public signing key objects.

Note:

Always returns at least POK. {free_ed25519_key_chain}

Definition at line 4547 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.42 ED25519_KEY dime_sgnt_signkeys_tls_fetch (signet_t const * *signet*)**

sets the signing key (pok - primary signing key in case of an org signet).

Parameters:

signet pointer to the target signet.

key public signing key to be set as the signing key of the signet.

format format specifier byte, dictating the format.

Returns:

0 on success, -1 on failure.

Definition at line 4564 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.43 int dime_sgnt_sok_create (signet_t * *signet*, ED25519_KEY * *key*, unsigned char *format*, uint8_t *perm*)

adds a sok (secondary organizational signing key) to an organizational signet.

Parameters:

signet pointer to the target org signet.

key ed25519 key to be added as a sok to the signet.

format format specifier byte dictating the format.

perm permissions for the usage of the sok.

Returns:

0 on success, -1 on failure.

Definition at line 4583 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.44 ED25519_KEY* dime_sgnt_sok_num_fetch (signet_t const * *signet*, unsigned int *num*)

fetch the secondary organizational signing key from the signet by number (starting at 1)

Parameters:

signet pointer to the target organizational signet.

num the sok number to be fetched.

Returns:

retrieved ed25519 key. {free_ed25519_key}

Definition at line 4600 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.45 char const* dime_sgnt_state_to_str (signet_state_t *state*)

returns a string from a signet_state_t enum type.

Parameters:

state signet state.

Returns:

null terminated string corresponding to the state.

Definition at line 4613 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.46 signet_type_t dime_sgnt_type_get (const signet_t * *signet*)

retrieves the signet type, org or user (signet_type_org or signet_type_user)

Parameters:

signet pointer to the target signet.

Returns:

a signet_type_t enum type with the signet type, signet_type_error on failure.

Definition at line 4627 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.47 int dime_sgnt_type_set (signet_t * *signet*, signet_type_t *type*)

sets the target signet to a specified type.

Parameters:

signet pointer to the target signet.

type specified signet type.

Returns:

0 on success, -1 on error.

Definition at line 4641 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.362.1.48 signet_state_t dime_sgnt_validate_all (signet_t const * *signet*, signet_t const * *previous*, signet_t const * *orgsig*, unsigned char const ** *dime_pok*)

verifies a user signet, org signet or ssr for syntax, context and cryptographic validity. does not perform chain of custody validation.

Parameters:

signet pointer to the target [signet_t](#) structure.

orgsig pointer to the org signet associated with the target signet if the target signet is a user signet. if target signet is not a user signet, orgsig should be passed as null.

previous pointer to the previous user signet, which will be used validate the chain of custody signature, if such is available.

dime_pok a null terminated array of pointers to ed25519 poks from the dime record associated with the target signet if the target signet is an org signet. if the target signet is not an org signet dime_pok should be passed as null;

Returns:

signet state as a signet_state_t enum type. ss_unknown on error.

Definition at line 4666 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

Referenced by _get_signet().

5.363 src/providers/dime/signet/signet.h File Reference

```
#include <stdint.h>
#include "dime/signet/common.h"
```

Data Structures

- struct [signet_t](#)

Enumerations

- enum [signet_state_t](#) {
[SS_UNKNOWN](#) = 0, [SS_MALFORMED](#), [SS_OVERFLOW](#), [SS_INCOMPLETE](#),
[SS_BROKEN_COC](#), [SS_INVALID](#), [SS_SSR](#), [SS_CRYPTO](#),
[SS_FULL](#), [SS_ID](#) }
- enum [signet_type_t](#) { [SIGNET_TYPE_ERROR](#), [SIGNET_TYPE_ORG](#) = 1, [SIGNET_TYPE_USER](#), [SIGNET_TYPE_SSR](#) }

Functions

- [EC_KEY *dime_sgnt_enckey_fetch](#) (const [signet_t](#) *signet)
Retrieves the public encryption key from the signet, if the signet is a user signet only retrieves the main encryption key (not alternate).
- [int dime_sgnt_enckey_set](#) ([signet_t](#) *signet, [EC_KEY](#) *key, unsigned char format)
Sets the public encryption key (non-alterante encryption key) for the signet.
- [int dime_sgnt_fid_count_get](#) (const [signet_t](#) *signet, unsigned char fid)
Retrieves the number of fields with the specified field id.
- [int dime_sgnt_fid_exists](#) (const [signet_t](#) *signet, unsigned char fid)
Checks for presence of field with specified id in the signet.
- [unsigned char *dime_sgnt_fid_num_fetch](#) (const [signet_t](#) *signet, unsigned char fid, unsigned int num, [size_t](#) *data_size)
Fetches the binary data value of the field specified by field id and the number at which it appears in the signet amongst fields with the same field id (1, 2, ...).
- [int dime_sgnt_fid_num_remove](#) ([signet_t](#) *signet, unsigned char fid, int num)
Removes the field specified by a field id and the number in which it appears in the target signet amongst fields with the same field id from the target signet.
- [int dime_sgnt_field_defined_create](#) ([signet_t](#) *signet, unsigned char fid, [size_t](#) data_size, const unsigned char *data)
Adds a field to the target field.
- [int dime_sgnt_field_defined_set](#) ([signet_t](#) *signet, unsigned char fid, [size_t](#) data_size, const unsigned char *data)
Replaces all fields in the target signet with the specified field id with a new field specified by the parameters.
- [int dime_sgnt_field_undefined_create](#) ([signet_t](#) *signet, [size_t](#) name_size, const unsigned char *name, [size_t](#) data_size, const unsigned char *data)
Adds an undefined field to signet with specified name and data.
- [unsigned char *dime_sgnt_field_undefined_fetch](#) (const [signet_t](#) *signet, [size_t](#) name_size, const unsigned char *name, [size_t](#) *data_size)

Fetches the first undefined field with the specified field name.

- int `dime_sgnt_field_undefined_remove` (`signet_t` *signet, size_t name_size, const unsigned char *name)
Removes an undefined field from the target signet by name.
- int `dime_sgnt_file_create` (`signet_t` *signet, const char *filename)
Stores a signet from the `signet_t` structure in a PEM formatted file specified by the filename.
- char * `dime_sgnt_fingerprint_crypto` (const `signet_t` *signet)
Takes a SHA512 fingerprint of a signet with all fields after the cryptographic signature field stripped off.
- char * `dime_sgnt_fingerprint_full` (const `signet_t` *signet)
Takes a SHA512 fingerprint of the user or org signet with the ID and FULL signature fields stripped off.
- char * `dime_sgnt_fingerprint_id` (const `signet_t` *signet)
Takes a SHA512 fingerprint of the entire user or org signet.
- char * `dime_sgnt_fingerprint_ssr` (const `signet_t` *signet)
Takes a SHA512 fingerprint of a user signet or an ssr with all fields after the SSR signature stripped off.
- char * `dime_sgnt_id_fetch` (`signet_t` *signet)
Retrieves signet id.
- int `dime_sgnt_id_set` (`signet_t` *signet, size_t id_size, const unsigned char *id)
Sets the ID of the signet to the specified NULL terminated string.
- int `dime_sgnt_msg_sig_verify` (const `signet_t` *signet, `ed25519_signature` sig, const unsigned char *buf, size_t buf_len)
Uses a signet's signing keys to verify a signature.
- int `dime_sgnt_sig_coc_sign` (`signet_t` *signet, `ED25519_KEY` *key)
Checks for the presence of all required fields that come before the chain of custody signature field and adds the SSR signature.
- int `dime_sgnt_sig_crypto_sign` (`signet_t` *signet, `ED25519_KEY` *key)
Signs an SSR or an incomplete ORG signet with the cryptographic signature after checking for the presence of all previous required fields.
- int `dime_sgnt_sig_full_sign` (`signet_t` *signet, `ED25519_KEY` *key)
Checks for the presence of all required fields that come before the full signature and signs all the fields that come before the CORE signature field.
- int `dime_sgnt_sig_id_sign` (`signet_t` *signet, `ED25519_KEY` *key)
Checks for the presence of all required fields that come before the FULL signature and signs the entire target signet using the specified key.
- int `dime_sgnt_sig_ssr_sign` (`signet_t` *signet, `ED25519_KEY` *key)
Checks for the presence of all required fields that come before the SSR signature field and adds the SSR signature.
- `signet_t` * `dime_sgnt_signet_binary_deserialize` (const unsigned char *in, size_t in_len)
Returns a new `signet_t` structure that gets deserialized from a data buffer.
- unsigned char * `dime_sgnt_signet_binary_serialize` (`signet_t` *signet, uint32_t *serial_size)
Serializes a signet structure into binary data.
- `signet_t` * `dime_sgnt_signet_b64_deserialize` (const char *b64_in)
Deserializes a b64 signet into a signet structure.

- `char * dime_sgnt_signet_b64_serialize (signet_t *signet)`
Serializes a signet structure into b64 data.
- `signet_t * dime_sgnt_signet_create (signet_type_t type)`
Returns a new `signet_t` structure.
- `signet_t * dime_sgnt_signet_create_w_keys (signet_type_t type, const char *keysfile)`
Creates a signet structure with public signing and encryption keys. Also creates a keys file in which the private keys are stored.
- `signet_t * dime_sgnt_signet_crypto_split (const signet_t *signet)`
Creates a copy of the target user signet with all fields beyond the INITIAL signature stripped off.
- `void dime_sgnt_signet_destroy (signet_t *signet)`
Destroys a `signet_t` structure.
- `void dime_sgnt_signet_dump (FILE *fp, signet_t *signet)`
Dumps signet into the specified file descriptor.
- `signet_t * dime_sgnt_signet_dupe (signet_t *signet)`
Create a copy of the provided signet.
- `signet_t * dime_sgnt_signet_full_split (const signet_t *signet)`
Creates a copy of the target signet with the ID field and the FULL signature stripped off.
- `signet_t * dime_sgnt_signet_load (const char *filename)`
Loads `signet_t` structure from a PEM formatted file specified by filename.
- `ED25519_KEY * dime_sgnt_signkey_fetch (const signet_t *signet)`
Retrieves the public signing key from the signet, if the signet is an org signet only retrieves the POK.
- `ED25519_KEY ** dime_sgnt_signkeys_msg_fetch (const signet_t *signet)`
Retrieves all the signing keys from an org signet that can be used to sign a signet.
- `ED25519_KEY ** dime_sgnt_signkeys_signet_fetch (const signet_t *signet)`
Retrieves all the signing keys from an org signet that can be used to sign software.
- `ED25519_KEY ** dime_sgnt_signkeys_software_fetch (const signet_t *signet)`
Retrieves all the signing keys from an org signet that can be used to sign a TLS certificate.
- `ED25519_KEY ** dime_sgnt_signkeys_tls_fetch (const signet_t *signet)`
sets the signing key (pok - primary signing key in case of an org signet).
- `int dime_sgnt_signkey_set (signet_t *signet, ED25519_KEY *key, unsigned char format)`
Retrieves all the signing keys from an org signet that can be used to sign a message.
- `int dime_sgnt_sok_create (signet_t *signet, ED25519_KEY *key, unsigned char format, uint8_t perm)`
adds a sok (secondary organizational signing key) to an organizational signet.
- `ED25519_KEY * dime_sgnt_sok_num_fetch (const signet_t *signet, unsigned int num)`
fetch the secondary organizational signing key from the signet by number (starting at 1)
- `const char * dime_sgnt_state_to_str (signet_state_t state)`
returns a string from a `signet_state_t` enum type.

- `signet_type_t dime_sgmt_type_get (const signet_t *signet)`
retrieves the signet type, org or user (signet_type_org or signet_type_user)
- `int dime_sgmt_type_set (signet_t *signet, signet_type_t type)`
sets the target signet to a specified type.
- `signet_state_t dime_sgmt_validate_all (const signet_t *signet, const signet_t *previous, const signet_t *orgsig, const unsigned char **dime_pok)`
verifies a user signet, org signet or ssr for syntax, context and cryptographic validity. does not perform chain of custody validation.

5.363.1 Enumeration Type Documentation

5.363.1.1 enum signet_state_t

Enumerator:

SS_UNKNOWN Invalid signet, state unknown/currently unclassified
SS_MALFORMED Invalid signet, it either doesn't fit the field format or has multiple unique fields
SS_OVERFLOW Invalid signet due to it being too large.
SS_INCOMPLETE Invalid signet, it is missing fields required to fit one of the valid categories, likely unsigned
SS_BROKEN_COC Invalid signet due to chain of custody signature being invalid
SS_INVALID Invalid signet, one or more signatures can not be verified
SS_SSR Valid unsigned SSR
SS_CRYPTO Valid cryptographic signet
SS_FULL Valid full signet
SS_ID Valid full signet with ID and organizational-identifiable-signature

Definition at line 7 of file signet.h.

5.363.1.2 enum signet_type_t

Enumerator:

SIGNET_TYPE_ERROR
SIGNET_TYPE_ORG
SIGNET_TYPE_USER
SIGNET_TYPE_SSR

Definition at line 20 of file signet.h.

5.363.2 Function Documentation

5.363.2.1 EC_KEY* dime_sgmt_enckey_fetch (signet_t const * signet)

Retrieves the public encryption key from the signet, if the signet is a user signet only retrieves the main encryption key (not alternate).

Parameters:

signet Pointer to the target signet.

Returns:

Pointer to the target encryption public key. {free_ec_key}

Definition at line 3909 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.2 int dime_sgnt_enckey_set (signet_t * *signet*, EC_KEY * *key*, unsigned char *format*)

Sets the public encryption key (non-alterante encryption key) for the signet.

Parameters:

signet Target signet.

key Public encryption key.

format Format specifier. TODO currently unused! (spec requires 0x04 but openssl natively serializes it to 0x02).

Returns:

0 on success, -1 on failure.

Definition at line 3926 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.3 int dime_sgnt_fid_count_get (signet_t const * *signet*, unsigned char *fid*)

Retrieves the number of fields with the specified field id.

Parameters:

signet Pointer to the target signet.

fid The target field id.

Returns:

The number of fields with specified field id. On various errors returns -1. NOTE: int overflow should not occur because of field size lower and signet size upper bounds.

Definition at line 3942 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.4 int dime_sgnt_fid_exists (signet_t const * *signet*, unsigned char *fid*)

Checks for presence of field with specified id in the signet.

Parameters:

signet The signet to be checked

fid Specified field id

Returns:

1 if such a field exists, 0 if it does not exist, -1 if error.

Definition at line 3956 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.5 unsigned char* dime_sgnt_fid_num_fetch (signet_t const * *signet*, unsigned char *fid*, unsigned int *num*, size_t * *data_size*)

Fetches the binary data value of the field specified by field id and the number at which it appears in the signet amongst fields with the same field id (1, 2, ...).

Parameters:

signet Pointer to the target signet.

fid Specified field id.

num Specified field number based on the order in which it appears in the signet.

data_size Pointer to the length of returned array.

Returns:

Array containing the binary data of the specified field, NULL if an error occurs. {free}

Definition at line 3978 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.6 int dime_sgnt_fid_num_remove (signet_t * *signet*, unsigned char *fid*, int *num*)

Removes the field specified by a field id and the number in which it appears in the target signet amongst fields with the same field id from the target signet.

Parameters:

signet Pointer to the target signet.

fid Field id of the field to be removed.

num The number in which the field to be removed appears amongst other fields with the same field id in the target signet, (1, 2, ...).

Returns:

0 on success, -1 on failure.

Definition at line 3996 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.7 int dime_sgnt_field_defined_create (signet_t * *signet*, unsigned char *fid*, size_t *data_size*, unsigned char const * *data*)

Adds a field to the target field.

Parameters:

signet Pointer to the target signet.

fid Field id of the field to be added.

data_size Size of the array containing the field data.

data Field data.

Returns:

0 on success, -1 on failure.

Definition at line 4012 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.8 `int dime_sgnt_field_defined_set (signet_t * signet, unsigned char fid, size_t data_size, const unsigned char * data)`

Replaces all fields in the target signet with the specified field id with a new field specified by the parameters.

Parameters:

- signet* Pointer to the target [signet_t](#) structure.
- fid* Field id which specifies the fields to be replaced with the new field.
- data_size* Size of field data array.
- data* Array containing field data.

Returns:

0 on success, -1 on failure.

Definition at line 4030 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.363.2.9 `int dime_sgnt_field_undefined_create (signet_t * signet, size_t name_size, unsigned char const * name, size_t data_size, unsigned char const * data)`

Adds an undefined field to signet with specified name and data.

Parameters:

- signet* Pointer to the target signet to which the field is added.
- name_size* Size of field name.
- name* Pointer to field name.
- data_size* Size of field data.
- data* Pointer to field data.

Returns:

0 on success, -1 on failure.

Definition at line 4050 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.363.2.10 `unsigned char* dime_sgnt_field_undefined_fetch (signet_t const * signet, size_t name_size, unsigned char const * name, size_t * data_size)`

Fetches the first undefined field with the specified field name.

Parameters:

- signet* Pointer to the target signet.
- name_size* Length of the passed array containing the length of the target field name.
- name* Array containing the name of the desired undefined field.
- data_size* Pointer to the size of the array that gets returned by the function.

Returns:

The array containing the data from the specified field or NULL in case of failure such as if the field was not found. {free}

Definition at line 4069 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.363.2.11 int dime_sgnt_field_undefined_remove (signet_t * *signet*, size_t *name_size*, unsigned char const * *name*)

Removes an undefined field from the target signet by name.

Parameters:

signet Pointer to the target signet.

name_size Size of field name.

name Name of the field to be removed.

Returns:

0 on success, -1 on failure.

Definition at line 4085 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.12 int dime_sgnt_file_create (signet_t * *signet*, char const * *filename*)

Stores a signet from the [signet_t](#) structure in a PEM formatted file specified by the filename.

Parameters:

signet Pointer to the [signet_t](#) structure containing the signet.

filename NULL terminated string containing the desired filename for the signet.

Returns:

0 on success, -1 on failure.

Definition at line 4100 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.13 char* dime_sgnt_fingerprint_crypto (signet_t const * *signet*)

Takes a SHA512 fingerprint of a signet with all fields after the cryptographic signature field stripped off.

Note:

To take an SSR fingerprint, use the `signet_ssr_fingerprint()` function.

Parameters:

signet Pointer to the target signet.

Returns:

Allocated NULL terminated string to a base64 encoded unpadded fingerprint. NULL on error. {free}

Definition at line 4118 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.14 char* dime_sgnt_fingerprint_full (signet_t const * *signet*)

Takes a SHA512 fingerprint of the user or org signet with the ID and FULL signature fields stripped off.

Note:

To take an SSR fingerprint, use the `signet_ssr_fingerprint()` function.

Parameters:

signet Pointer to the target signet.

Returns:

Allocated NULL terminated buffer to a base64 encoded unpadded fingerprint. NULL on failure. {free}

Definition at line 4136 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.363.2.15 char* dime_sgnt_fingerprint_id (signet_t const * *signet*)

Takes a SHA512 fingerprint of the entire user or org signet.

Parameters:

signet Pointer to the target signet.

Returns:

Allocated NULL terminated buffer to a base64 encoded unpadded fingerprint. NULL on failure; {free}

Definition at line 4151 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.363.2.16 char* dime_sgnt_fingerprint_ssr (signet_t const * *signet*)

Takes a SHA512 fingerprint of a user signet or an ssr with all fields after the SSR signature stripped off.

Parameters:

signet Pointer to the target signet.

Returns:

Allocated NULL terminated buffer to a base64 encoded unpadded fingerprint. {free}

Definition at line 4166 of file `signet.c`.

References `PUBLIC_FUNCTION_IMPLEMENT`.

5.363.2.17 char* dime_sgnt_id_fetch (signet_t * *signet*)

Retrieves signet id.

Parameters:

signet Signet from which the id is retrieved.

Returns:

Cstring containing ID or NULL on error. {free}

Definition at line 4180 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.18 int dime_sgnt_id_set (signet_t * *signet*, size_t *id_size*, unsigned char const * *id*)

Sets the ID of the signet to the specified NULL terminated string.

Parameters:

signet Pointer to the target signet.

id_size Size of signet id.

id Signet id.

Returns:

0 on success, -1 on failure.

Definition at line 4196 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.19 int dime_sgnt_msg_sig_verify (signet_t const * *signet*, ed25519_signature *sig*, unsigned char const * *buf*, size_t *buf_len*)

Uses a signet's signing keys to verify a signature.

Parameters:

signet Pointer to the signet.

sig ed25519 signature buffer to be verified.

buf Data buffer over which the signature was taken.

buf_len Length of data buffer.

Returns:

1 on successful verification, 0 if the signature could not be verified, -1 if an error occurred.

Definition at line 4215 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.20 int dime_sgnt_sig_coc_sign (signet_t * *signet*, ED25519_KEY * *key*)

Checks for the presence of all required fields that come before the chain of custody signature field and adds the SSR signature.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4230 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.21 int dime_sgmt_sig_crypto_sign (signet_t * *signet*, ED25519_KEY * *key*)

Signs an SSR or an incomplete ORG signet with the cryptographic signature after checking for the presence of all previous required fields.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4245 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.22 int dime_sgmt_sig_full_sign (signet_t * *signet*, ED25519_KEY * *key*)

Checks for the presence of all required fields that come before the full signature and signs all the fields that come before the CORE signature field.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4261 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.23 int dime_sgmt_sig_id_sign (signet_t * *signet*, ED25519_KEY * *key*)

Checks for the presence of all required fields that come before the FULL signature and signs the entire target signet using the specified key.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4275 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.24 int dime_sgmt_sig_ssr_sign (signet_t * *signet*, ED25519_KEY * *key*)

Checks for the presence of all required fields that come before the SSR signature field and adds the SSR signature.

Parameters:

signet Pointer to the target [signet_t](#) structure.

key Specified ed25519 key used for signing.

Returns:

0 on success, -1 on failure.

Definition at line 4290 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.25 `signet_t* dime_sgnt_signet_b64_deserialize (const char * b64_in)`

Deserializes a b64 signet into a signet structure.

Parameters:

b64_in NULL terminated array of b64 signet data.

Returns:

Pointer to newly allocated signet structure, NULL if failure. {dime_sgnt_destroy_signet}

Definition at line 4336 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

Referenced by `_get_signet()`.

5.363.2.26 `char* dime_sgnt_signet_b64_serialize (signet_t * signet)`

Serializes a signet structure into b64 data.

Parameters:

signet Pointer to the target signet.

Returns:

Signet serialized into b64 data. NULL on error. {free}

Definition at line 4350 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.27 `signet_t* dime_sgnt_signet_binary_deserialize (unsigned char const * in, size_t len)`

Returns a new [signet_t](#) structure that gets deserialized from a data buffer.

Parameters:

in data buffer containing the binary form of a signet

in_len length of data buffer

Returns:

A pointer to a newly allocated [signet_t](#) structure type, NULL on failure. {dime_sgnt_destroy_signet}

Definition at line 4306 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

Referenced by `_deserialize_signet_cb()`.

5.363.2.28 unsigned char* dime_sgnt_signet_binary_serialize (signet_t * *signet*, uint32_t * *serial_size*)

Serializes a signet structure into binary data.

Parameters:

signet Pointer to the target signet.

serial_size Pointer to the value that stores the length of the array returned.

Returns:

Signet serialized into binary data. NULL on error. {free}

Definition at line 4322 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

Referenced by _serialize_signet_cb().

5.363.2.29 signet_t* dime_sgnt_signet_create (signet_type_t *type*)

Returns a new [signet_t](#) structure.

Parameters:

type signet type user org or sss (SIGNET_TYPE_USER, SIGNET_TYPE_ORG or SIGNET_TYPE_SSR)

Returns:

A pointer to a newly allocated [signet_t](#) structure type, NULL if failure. {dime_sgnt_destroy_signet}

Definition at line 4364 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.30 signet_t* dime_sgnt_signet_create_w_keys (signet_type_t *type*, char const * *keyfile*)

Creates a signet structure with public signing and encryption keys. Also creates a keys file in which the private keys are stored.

Parameters:

type Signet type, org, user or sss (SIGNET_TYPE_ORG, SIGNET_TYPE_USER or SIGNET_TYPE_SSR).

keyfile NULL terminated string containing the name of the keyfile to be created.

Returns:

Pointer to the newly created and allocated [signet_t](#) structure or NULL on error. {dime_sgnt_destroy_signet}

Definition at line 4383 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.31 signet_t* dime_sgnt_signet_crypto_split (signet_t const * *signet*)

Creates a copy of the target user signet with all fields beyond the INITIAL signature stripped off.

Parameters:

signet Pointer to the target signet.

Returns:

Pointer to a stripped signet on success, NULL on failure. {dime_sgnt_destroy_signet}

Definition at line 4398 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.32 void dime_sgnt_signet_destroy (signet_t * *signet*)

Destroys a [signet_t](#) structure.

Parameters:

signet Pointer to the signet to be destroyed.

Definition at line 4408 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

Referenced by _destroy_signet_cb(), and _get_signet().

5.363.2.33 void dime_sgnt_signet_dump (FILE * *fp*, signet_t * *signet*)

Dumps signet into the specified file descriptor.

Parameters:

fp File descriptor the signet is dumped to.

signet Pointer to the [signet_t](#) structure to be dumped.

Definition at line 4420 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT_VOID.

Referenced by _dump_signet_cb().

5.363.2.34 signet_t* dime_sgnt_signet_dupe (signet_t * *signet*)

Create a copy of the provided signet.

Parameters:

signet Signet to be copied.

Returns:

The copy. {dime_sgnt_destroy_signet}

Definition at line 4434 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.35 signet_t* dime_sgnt_signet_full_split (signet_t const * *signet*)

Creates a copy of the target signet with the ID field and the FULL signature stripped off.

Parameters:

signet Pointer to the target signet.

Returns:

Pointer to a stripped signet on success, NULL on failure. {dime_sgmt_destroy_signet}

Definition at line 4449 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.36 signet_t* dime_sgmt_signet_load (char const *filename)

Loads [signet_t](#) structure from a PEM formatted file specified by filename.

Parameters:

filename NULL terminated string containing the filename of the file containing the signet.

Returns:

Pointer to a newly created [signet_t](#) structure loaded from the file, NULL on failure. {dime_sgmt_destroy_signet}

Definition at line 4465 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.37 ED25519_KEY* dime_sgmt_signkey_fetch (signet_t const *signet)

Retrieves the public signing key from the signet, if the signet is an org signet only retrieves the POK.

Parameters:

signet Pointer to the target signet.

Returns:

Pointer to the target ed25519 public key. {free_ed25519_key}

Definition at line 4480 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.38 int dime_sgmt_signkey_set (signet_t *signet, ED25519_KEY *key, unsigned char format)

Retrieves all the signing keys from an org signet that can be used to sign a message.

Parameters:

signet Pointer to target organizational signet.

Returns:

A NULL pointer terminated array of ed25519 public signing key objects.

Note:

Always returns at least POK. {free_ed25519_key_chain}

Definition at line 4496 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.39 ED25519_KEY dime_sgnt_signkeys_msg_fetch (signet_t const * *signet*)**

Retrieves all the signing keys from an org signet that can be used to sign a signet.

Parameters:

signet Pointer to target organizational signet.

Returns:

A NULL pointer terminated array of ed25519 public signing key objects.

Note:

Always returns at least POK. {free_ed25519_key_chain}

Definition at line 4513 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.40 ED25519_KEY dime_sgnt_signkeys_signet_fetch (signet_t const * *signet*)**

Retrieves all the signing keys from an org signet that can be used to sign software.

Parameters:

signet Pointer to target organizational signet.

Returns:

A NULL pointer terminated array of ed25519 public signing key objects.

Note:

Always returns at least POK. {free_ed25519_key_chain}

Definition at line 4530 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.41 ED25519_KEY dime_sgnt_signkeys_software_fetch (signet_t const * *signet*)**

Retrieves all the signing keys from an org signet that can be used to sign a TLS certificate.

Parameters:

signet Pointer to target organizational signet.

Returns:

A NULL pointer terminated array of ed25519 public signing key objects.

Note:

Always returns at least POK. {free_ed25519_key_chain}

Definition at line 4547 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.42 ED25519_KEY dime_sgnt_signkeys_tls_fetch (signet_t const * *signet*)**

sets the signing key (pok - primary signing key in case of an org signet).

Parameters:

signet pointer to the target signet.

key public signing key to be set as the signing key of the signet.

format format specifier byte, dictating the format.

Returns:

0 on success, -1 on failure.

Definition at line 4564 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.43 int dime_sgnt_sok_create (signet_t * *signet*, ED25519_KEY * *key*, unsigned char *format*, uint8_t *perm*)

adds a sok (secondary organizational signing key) to an organizational signet.

Parameters:

signet pointer to the target org signet.

key ed25519 key to be added as a sok to the signet.

format format specifier byte dictating the format.

perm permissions for the usage of the sok.

Returns:

0 on success, -1 on failure.

Definition at line 4583 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.44 ED25519_KEY* dime_sgnt_sok_num_fetch (signet_t const * *signet*, unsigned int *num*)

fetch the secondary organizational signing key from the signet by number (starting at 1)

Parameters:

signet pointer to the target organizational signet.

num the sok number to be fetched.

Returns:

retrieved ed25519 key. {free_ed25519_key}

Definition at line 4600 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.45 `const char* dime_sgnt_state_to_str (signet_state_t state)`

returns a string from a signet_state_t enum type.

Parameters:

state signet state.

Returns:

null terminated string corresponding to the state.

Definition at line 4613 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.46 `signet_type_t dime_sgnt_type_get (const signet_t * signet)`

retrieves the signet type, org or user (signet_type_org or signet_type_user)

Parameters:

signet pointer to the target signet.

Returns:

a signet_type_t enum type with the signet type, signet_type_error on failure.

Definition at line 4627 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.47 `int dime_sgnt_type_set (signet_t * signet, signet_type_t type)`

sets the target signet to a specified type.

Parameters:

signet pointer to the target signet.

type specified signet type.

Returns:

0 on success, -1 on error.

Definition at line 4641 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

5.363.2.48 `signet_state_t dime_sgnt_validate_all (signet_t const * signet, signet_t const * previous, signet_t const * orgsig, unsigned char const ** dime_pok)`

verifies a user signet, org signet or ssr for syntax, context and cryptographic validity. does not perform chain of custody validation.

Parameters:

signet pointer to the target [signet_t](#) structure.

orgsig pointer to the org signet associated with the target signet if the target signet is a user signet. if target signet is not a user signet, orgsig should be passed as null.

previous pointer to the previous user signet, which will be used validate the chain of custody signature, if such is available.

dime_pok a null terminated array of pointers to ed25519 poks from the dime record associated with the target signet if the target signet is an org signet. if the target signet is not an org signet dime_pok should be passed as null;

Returns:

signet state as a signet_state_t enum type. ss_unknown on error.

Definition at line 4666 of file signet.c.

References PUBLIC_FUNCTION_IMPLEMENT.

Referenced by _get_signet().

5.364 src/providers/dime/util/encoding.c File Reference

```
#include "dime/util/encoding.h"
#include <string.h>
```

Functions

- `derror_t` const * `libdime_base64_decode` (`dime_ctx_t` const * `dime_ctx`, unsigned char **`result`, `size_t` *`result_length`, char const *`buf`, `size_t` `len`)
Perform base64-decoding on a string and return the result.
- `derror_t` const * `libdime_base64_encode` (`dime_ctx_t` const * `dime_ctx`, `sds` *`result`, unsigned char const *`buf`, `size_t` `len`)
Return the base64-encoded string of a data buffer.

5.364.1 Function Documentation

5.364.1.1 `derror_t` const* `libdime_base64_decode` (`dime_ctx_t` const * `dime_ctx`, unsigned char ** `result`, `size_t` * `result_length`, char const * `buf`, `size_t` `len`)

Perform base64-decoding on a string and return the result.

Note:

SOLVED PROBLEM - why are we reinventing the wheel?

Parameters:

- result*** double pointer to a newly allocated buffer containing the base64-decoded data, or NULL on failure.
- buf*** a pointer to a buffer containing the data to be decoded.
- len*** the length, in bytes, of the buffer to be decoded.
- outlen*** a pointer to a variable to receive the length of the decoded data. {free}

Definition at line 36 of file encoding.c.

References `DIME_LOG_ERROR`, `ERR_BAD_PARAM`, `ERR_ENCODING`, and `ERR_NOMEM`.

5.364.1.2 `derror_t` const* `libdime_base64_encode` (`dime_ctx_t` const * `dime_ctx`, `sds` * `result`, unsigned char const * `buf`, `size_t` `len`)

Return the base64-encoded string of a data buffer.

Parameters:

- result*** a pointer to a newly allocated sds string containing the base64-encoded data, or NULL on failure.
- buf*** a pointer to the data buffer to be base-64 encoded.
- len*** the length, in bytes, of the buffer to be encoded. {sdsfree}

Definition at line 155 of file encoding.c.

References `DIME_LOG_ERROR`, `ERR_BAD_PARAM`, `ERR_NOMEM`, `sdsfree()`, and `sdsnewlen()`.

5.365 src/providers/dime/util/encoding.h File Reference

```
#include "dime/dime_ctx.h"
#include "dime/error_codes.h"
#include "dime/sds/sds.h"
#include <stddef.h>
#include <stdlib.h>
```

Functions

- `derror_t` const * `libdime_base64_decode` (`dime_ctx_t` const * `dime_ctx`, unsigned char **`result`, `size_t` *`result_length`, char const *`buf`, `size_t` `len`)
Perform base64-decoding on a string and return the result.
- `derror_t` const * `libdime_base64_encode` (`dime_ctx_t` const * `dime_ctx`, `sds` *`result`, unsigned char const *`buf`, `size_t` `len`)
Return the base64-encoded string of a data buffer.

5.365.1 Function Documentation

5.365.1.1 `derror_t` const* `libdime_base64_decode` (`dime_ctx_t` const * `dime_ctx`, unsigned char ** `result`, `size_t` * `result_length`, char const * `buf`, `size_t` `len`)

Perform base64-decoding on a string and return the result.

Note:

SOLVED PROBLEM - why are we reinventing the wheel?

Parameters:

result double pointer to a newly allocated buffer containing the base64-decoded data, or NULL on failure.

buf a pointer to a buffer containing the data to be decoded.

len the length, in bytes, of the buffer to be decoded.

outlen a pointer to a variable to receive the length of the decoded data. {free}

Definition at line 36 of file encoding.c.

References DIME_LOG_ERROR, ERR_BAD_PARAM, ERR_ENCODING, and ERR_NOMEM.

5.365.1.2 `derror_t` const* `libdime_base64_encode` (`dime_ctx_t` const * `dime_ctx`, `sds` * `result`, unsigned char const * `buf`, `size_t` `len`)

Return the base64-encoded string of a data buffer.

Parameters:

result a pointer to a newly allocated sds string containing the base64-encoded data, or NULL on failure.

buf a pointer to the data buffer to be base-64 encoded.

len the length, in bytes, of the buffer to be encoded. {sdsfree}

Definition at line 155 of file encoding.c.

References DIME_LOG_ERROR, ERR_BAD_PARAM, ERR_NOMEM, sdsfree(), and sdsnewlen().

5.366 src/providers/dime/util/encrypt.c File Reference

```
#include "dime/util/encrypt.h"
#include <stdio.h>
#include <string.h>
#include <openssl/bio.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <openssl/rand.h>
```

Data Structures

- struct [encrypt_ctx](#)

Defines

- #define [LOG_OPENSSL_ERROR](#)(ctx, buf, msg)

Functions

- [derror_t](#) const * [encrypt_ctx_new](#) ([dime_ctx_t](#) const *[dime_ctx](#), [encrypt_ctx_t](#) **result)
Initialize the cryptographic subsystem.
- void [encrypt_ctx_free](#) ([encrypt_ctx_t](#) *ctx)
Shutdown the cryptographic subsystem.
- [derror_t](#) const * [encrypt_keypair_generate](#) ([dime_ctx_t](#) const *[dime_ctx](#), [encrypt_ctx_t](#) const *[encrypt_ctx](#), [encrypt_keypair_t](#) **result)
Generate an EC key pair.
- [derror_t](#) const * [encrypt_deserialize_pubkey](#) ([dime_ctx_t](#) const *[dime_ctx](#), [encrypt_keypair_t](#) **result, unsigned char const *buf, size_t blen)
Deserialize an EC public key stored in binary format.
- [derror_t](#) const * [encrypt_deserialize_privkey](#) ([dime_ctx_t](#) const *[dime_ctx](#), [encrypt_keypair_t](#) **result, unsigned char const *buf, size_t blen)
Deserialize an EC private key stored in binary format.

5.366.1 Define Documentation

5.366.1.1 #define LOG_OPENSSL_ERROR(ctx, buf, msg)

Value:

```
do {\
    get_openssl_error(buf, sizeof(buf)); \
    DIME_LOG_ERROR(dime_ctx, buf); \
    DIME_LOG_ERROR(dime_ctx, msg); \
} while (0)
```

Definition at line 74 of file encrypt.c.

Referenced by `encrypt_ctx_new()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, and `encrypt_keypair_generate()`.

5.366.2 Function Documentation

5.366.2.1 `void encrypt_ctx_free (encrypt_ctx_t * ctx)`

Shutdown the cryptographic subsystem.

Definition at line 156 of file encrypt.c.

References `EC_GROUP_clear_free_d`, `encrypt_ctx::encryption_group`, `ERR_free_strings_d`, and `EVP_cleanup_d`.

5.366.2.2 `derror_t const* encrypt_ctx_new (dime_ctx_t const * dime_ctx, encrypt_ctx_t ** result)`

Initialize the cryptographic subsystem.

Note:

This is currently NOT reentrant. (Apparently OpenSSL likes global state :P)

Definition at line 88 of file encrypt.c.

References `EC_GROUP_clear_free_d`, `EC_GROUP_new_by_curve_name_d`, `ERR_BAD_PARAM`, `ERR_CRYPTO`, `ERR_free_strings_d`, `ERR_NOMEM`, `EVP_cleanup_d`, `EVP_get_digestbyname_d`, `LOG_OPENSSL_ERROR`, `OBJ_nid2sn_d`, `OPENSSL_add_all_algorithms_noconf_d`, `SSL_library_init_d`, and `SSL_load_error_strings_d`.

5.366.2.3 `derror_t const* encrypt_deserialize_privkey (dime_ctx_t const * dime_ctx, encrypt_keypair_t ** result, unsigned char const * buf, size_t blen)`

Deserialize an EC private key stored in binary format.

Parameters:

result a pointer to the deserialized EC private key on success, or NULL on failure.

buf a pointer to the buffer holding the EC private key in binary format.

blen the length, in bytes, of the buffer holding the EC private key.

Definition at line 306 of file encrypt.c.

References `d2i_ECPrivateKey_d`, `EC_GROUP_new_by_curve_name_d`, `EC_KEY_free_d`, `EC_KEY_new_d`, `EC_KEY_set_group_d`, `ERR_BAD_PARAM`, `ERR_CRYPTO`, and `LOG_OPENSSL_ERROR`.

5.366.2.4 `derror_t const* encrypt_deserialize_pubkey (dime_ctx_t const * dime_ctx, encrypt_keypair_t ** result, unsigned char const * buf, size_t blen)`

Deserialize an EC public key stored in binary format.

Parameters:

result a pointer to the deserialized EC public key on success, or NULL on failure.

buf a pointer to the buffer holding the EC public key in binary format.

blen the length, in bytes, of the buffer holding the EC public key.

Definition at line 235 of file encrypt.c.

References `EC_GROUP_new_by_curve_name_d`, `EC_KEY_free_d`, `EC_KEY_new_d`, `EC_KEY_set_group_d`, `ERR_BAD_PARAM`, `ERR_CRYPTO`, `LOG_OPENSSL_ERROR`, and `o2i_ECPublicKey_d`.

5.366.2.5 `derror_t` const* `encrypt_keypair_generate` (`dime_ctx_t` const * *dime_ctx*, `encrypt_ctx_t` const * *encrypt_ctx*, `encrypt_keypair_t` ** *result*)

Generate an EC key pair.

Parameters:

result a newly allocated and generated EC key pair on success, or NULL on failure.

Definition at line 175 of file `encrypt.c`.

References `EC_KEY_free_d`, `EC_KEY_generate_key_d`, `EC_KEY_new_d`, `EC_KEY_set_group_d`, `encrypt_ctx::encryption_group`, `ERR_BAD_PARAM`, `ERR_CRYPT`, and `LOG_OPENSSL_ERROR`.

5.367 src/providers/dime/util/encrypt.h File Reference

```
#include "dime/dime_ctx.h"
#include "dime/error_codes.h"
```

Data Structures

- struct [encrypt_keypair_t](#)

Typedefs

- typedef struct [encrypt_ctx](#) [encrypt_ctx_t](#)

Functions

- [derror_t](#) const * [encrypt_ctx_new](#) ([dime_ctx_t](#) const *[dime_ctx](#), [encrypt_ctx_t](#) **result)
Initialize the cryptographic subsystem.
- void [encrypt_ctx_free](#) ([encrypt_ctx_t](#) *ctx)
Shutdown the cryptographic subsystem.
- [derror_t](#) const * [encrypt_keypair_generate](#) ([dime_ctx_t](#) const *[dime_ctx](#), [encrypt_ctx_t](#) const *[encrypt_ctx](#), [encrypt_keypair_t](#) **result)
Generate an EC key pair.
- [derror_t](#) const * [encrypt_deserialize_pubkey](#) ([dime_ctx_t](#) const *[dime_ctx](#), [encrypt_keypair_t](#) **result, unsigned char const *buf, size_t blen)
Deserialize an EC public key stored in binary format.
- [derror_t](#) const * [encrypt_deserialize_privkey](#) ([dime_ctx_t](#) const *[dime_ctx](#), [encrypt_keypair_t](#) **result, unsigned char const *buf, size_t blen)
Deserialize an EC private key stored in binary format.

5.367.1 Typedef Documentation

5.367.1.1 typedef struct encrypt_ctx encrypt_ctx_t

Definition at line 7 of file encrypt.h.

5.367.2 Function Documentation

5.367.2.1 void encrypt_ctx_free (encrypt_ctx_t * ctx)

Shutdown the cryptographic subsystem.

Definition at line 156 of file encrypt.c.

References [EC_GROUP_clear_free_d](#), [encrypt_ctx::encryption_group](#), [ERR_free_strings_d](#), and [EVP_cleanup_d](#).

5.367.2.2 `derror_t` const* `encrypt_ctx_new` (`dime_ctx_t` const * *dime_ctx*, `encrypt_ctx_t` ** *result*)

Initialize the cryptographic subsystem.

Note:

This is currently NOT reentrant. (Apparently OpenSSL likes global state :P)

Definition at line 88 of file `encrypt.c`.

References `EC_GROUP_clear_free_d`, `EC_GROUP_new_by_curve_name_d`, `ERR_BAD_PARAM`, `ERR_CRYPT`, `ERR_free_strings_d`, `ERR_NOMEM`, `EVP_cleanup_d`, `EVP_get_digestbyname_d`, `LOG_OPENSSL_ERROR`, `OBJ_nid2sn_d`, `OPENSSL_add_all_algorithms_noconf_d`, `SSL_library_init_d`, and `SSL_load_error_strings_d`.

5.367.2.3 `derror_t` const* `encrypt_deserialize_privkey` (`dime_ctx_t` const * *dime_ctx*, `encrypt_keypair_t` ** *result*, unsigned char const * *buf*, size_t *blen*)

Deserialize an EC private key stored in binary format.

Parameters:

result a pointer to the deserialized EC private key on success, or NULL on failure.

buf a pointer to the buffer holding the EC private key in binary format.

blen the length, in bytes, of the buffer holding the EC private key.

Definition at line 306 of file `encrypt.c`.

References `d2i_ECPrivateKey_d`, `EC_GROUP_new_by_curve_name_d`, `EC_KEY_free_d`, `EC_KEY_new_d`, `EC_KEY_set_group_d`, `ERR_BAD_PARAM`, `ERR_CRYPT`, and `LOG_OPENSSL_ERROR`.

5.367.2.4 `derror_t` const* `encrypt_deserialize_pubkey` (`dime_ctx_t` const * *dime_ctx*, `encrypt_keypair_t` ** *result*, unsigned char const * *buf*, size_t *blen*)

Deserialize an EC public key stored in binary format.

Parameters:

result a pointer to the deserialized EC public key on success, or NULL on failure.

buf a pointer to the buffer holding the EC public key in binary format.

blen the length, in bytes, of the buffer holding the EC public key.

Definition at line 235 of file `encrypt.c`.

References `EC_GROUP_new_by_curve_name_d`, `EC_KEY_free_d`, `EC_KEY_new_d`, `EC_KEY_set_group_d`, `ERR_BAD_PARAM`, `ERR_CRYPT`, `LOG_OPENSSL_ERROR`, and `o2i_ECPublicKey_d`.

5.367.2.5 `derror_t` const* `encrypt_keypair_generate` (`dime_ctx_t` const * *dime_ctx*, `encrypt_ctx_t` const * *encrypt_ctx*, `encrypt_keypair_t` ** *result*)

Generate an EC key pair.

Parameters:

result a newly allocated and generated EC key pair on success, or NULL on failure.

Definition at line 175 of file `encrypt.c`.

References `EC_KEY_free_d`, `EC_KEY_generate_key_d`, `EC_KEY_new_d`, `EC_KEY_set_group_d`, `encrypt_ctx::encryption_group`, `ERR_BAD_PARAM`, `ERR_CRYPT`, and `LOG_OPENSSL_ERROR`.

5.368 src/providers/images/freetype.c File Reference

```
#include "magma.h"
```

Functions

- [chr_t * lib_version_freetype](#) (void)
Return the version string of the font library.
- [bool_t lib_load_freetype](#) (void)
Initialize the font library and bind dynamically to the exported functions that are required.

Variables

- void * [lib_magma](#)

5.368.1 Function Documentation

5.368.1.1 bool_t lib_load_freetype (void)

Initialize the font library and bind dynamically to the exported functions that are required. [freetype.c](#)

Returns:

true on success or false on failure.

Definition at line 37 of file [freetype.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.368.1.2 chr_t* lib_version_freetype (void)

Return the version string of the font library.

Returns:

a pointer to a character string containing the font library version information.

Definition at line 17 of file [freetype.c](#).

References [FT_Done_FreeType_d](#), [FT_Init_FreeType_d](#), and [FT_Library_Version_d](#).

Referenced by [lib_load\(\)](#).

5.368.2 Variable Documentation

5.368.2.1 void* lib_magma

Referenced by [lib_load\(\)](#), [lib_unload\(\)](#), and [STACK_OF\(\)](#).

5.369 src/providers/images/gd.c File Reference

```
#include "magma.h"
```

Functions

- [chr_t * lib_version_gd](#) (void)
Return the version string of the gd library.
- [bool_t lib_load_gd](#) (void)
Initialize the gd library and bind dynamically to the exported functions that are required.

5.369.1 Function Documentation

5.369.1.1 [bool_t lib_load_gd](#) (void)

Initialize the gd library and bind dynamically to the exported functions that are required. [gd.c](#)

Returns:

true on success or false on failure.

Definition at line 22 of file [gd.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.369.1.2 [chr_t* lib_version_gd](#) (void)

Return the version string of the gd library.

Returns:

a pointer to a character string containing the gd library version information.

Definition at line 14 of file [gd.c](#).

References [gdVersionString_d](#).

Referenced by [lib_load\(\)](#).

5.370 src/providers/images/images.h File Reference

Functions

- [bool_t lib_load_gd](#) (void)
[gd.c](#)
- [chr_t * lib_version_gd](#) (void)
Return the version string of the gd library.
- [bool_t lib_load_freetype](#) (void)
[freetype.c](#)
- [chr_t * lib_version_freetype](#) (void)
Return the version string of the font library.
- [bool_t lib_load_jpeg](#) (void)
[jpeg.c](#)
- [chr_t * lib_version_jpeg](#) (void)
Return the version string of the jpeg library.
- [bool_t lib_load_png](#) (void)
[png.c](#)
- [chr_t * lib_version_png](#) (void)
Return the version string of png library.

5.370.1 Function Documentation

5.370.1.1 bool_t lib_load_freetype (void)

[freetype.c](#) [freetype.c](#)

Returns:

true on success or false on failure.

Definition at line 37 of file [freetype.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.370.1.2 bool_t lib_load_gd (void)

[gd.c](#) [gd.c](#)

Returns:

true on success or false on failure.

Definition at line 22 of file [gd.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.370.1.3 `bool_t lib_load_jpeg (void)`

[jpeg.c jpeg.c](#)

Returns:

true on success or false on failure.

Definition at line 22 of file jpeg.c.

References `lib_symbols()`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.370.1.4 `bool_t lib_load_png (void)`

[png.c png.c](#)

Returns:

true on success or false on failure.

Definition at line 39 of file png.c.

References `lib_symbols()`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.370.1.5 `chr_t* lib_version_freetype (void)`

Return the version string of the font library.

Returns:

a pointer to a character string containing the font library version information.

Definition at line 17 of file freetype.c.

References `FT_Done_FreeType_d`, `FT_Init_FreeType_d`, and `FT_Library_Version_d`.

Referenced by `lib_load()`.

5.370.1.6 `chr_t* lib_version_gd (void)`

Return the version string of the gd library.

Returns:

a pointer to a character string containing the gd library version information.

Definition at line 14 of file gd.c.

References `gdVersionString_d`.

Referenced by `lib_load()`.

5.370.1.7 `chr_t* lib_version_jpeg (void)`

Return the version string of the jpeg library.

Returns:

a pointer to a character string containing the jpeg library version information.

Definition at line 14 of file jpeg.c.

References jpeg_version_d.

Referenced by lib_load().

5.370.1.8 chr_t* lib_version_png (void)

Return the version string of png library.

Returns:

a pointer to a character string containing the png library version information.

Definition at line 16 of file png.c.

References png_access_version_number_d.

Referenced by lib_load().

5.371 src/providers/images/jpeg.c File Reference

```
#include "magma.h"
```

Functions

- [chr_t * lib_version_jpeg](#) (void)
Return the version string of the jpeg library.
- [bool_t lib_load_jpeg](#) (void)
Initialize the jpeg library and bind dynamically to the exported functions that are required.

5.371.1 Function Documentation

5.371.1.1 bool_t lib_load_jpeg (void)

Initialize the jpeg library and bind dynamically to the exported functions that are required. [jpeg.c](#)

Returns:

true on success or false on failure.

Definition at line 22 of file jpeg.c.

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.371.1.2 chr_t* lib_version_jpeg (void)

Return the version string of the jpeg library.

Returns:

a pointer to a character string containing the jpeg library version information.

Definition at line 14 of file jpeg.c.

References [jpeg_version_d](#).

Referenced by [lib_load\(\)](#).

5.372 src/providers/images/png.c File Reference

```
#include "magma.h"
```

Functions

- [chr_t * lib_version_png](#) (void)
Return the version string of png library.
- [bool_t lib_load_png](#) (void)
Initialize the png library and bind dynamically to the exported functions that are required.

5.372.1 Function Documentation

5.372.1.1 [bool_t lib_load_png](#) (void)

Initialize the png library and bind dynamically to the exported functions that are required. [png.c](#)

Returns:

true on success or false on failure.

Definition at line 39 of file [png.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.372.1.2 [chr_t* lib_version_png](#) (void)

Return the version string of png library.

Returns:

a pointer to a character string containing the png library version information.

Definition at line 16 of file [png.c](#).

References [png_access_version_number_d](#).

Referenced by [lib_load\(\)](#).

5.373 src/providers/parsers/json.c File Reference

```
#include "magma.h"
```

Functions

- [chr_t * lib_version_jansson](#) (void)
Return the version string for libjansson.
- [bool_t lib_load_jansson](#) (void)
Initialize the Jansson library and dynamically bind to the required symbols.

5.373.1 Function Documentation

5.373.1.1 bool_t lib_load_jansson (void)

Initialize the Jansson library and dynamically bind to the required symbols. [json.c](#)

Returns:

true on success or false on failure.

Definition at line 22 of file json.c.

References [lib_symbols\(\)](#), [M_BIND](#), and [symbol_t](#).

Referenced by [lib_load\(\)](#).

5.373.1.2 chr_t* lib_version_jansson (void)

Return the version string for libjansson.

Returns:

a pointer to a character string containing the libjansson version information.

Definition at line 14 of file json.c.

References [jansson_version_d](#).

Referenced by [lib_load\(\)](#).

5.374 src/providers/parsers/utf8.c File Reference

```
#include "magma.h"
```

Functions

- `chr_t * lib_version_utf8proc` (void)
Return the shared library version string for libutf8proc.
- `bool_t lib_load_utf8proc` (void)
Initialize the UTF8 library and dynamically bind to the required symbols.
- `const chr_t * utf8_error_string` (ssize_t error_code)
Exchange an error code for a string explaining the nature of the UTF8 error.
- `size_t utf8_length_st` (stringer_t *s)
Determine how many valid Unicode characters the string has.
- `bool_t utf8_valid_st` (stringer_t *s)
Determine whether a managed string is a properly formatted UTF8 string.

5.374.1 Function Documentation

5.374.1.1 `bool_t lib_load_utf8proc` (void)

Initialize the UTF8 library and dynamically bind to the required symbols. utf.c

Returns:

true on success or false on failure.

Definition at line 23 of file utf8.c.

References `lib_symbols()`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.374.1.2 `chr_t* lib_version_utf8proc` (void)

Return the shared library version string for libutf8proc.

Returns:

a pointer to a character string containing the libutf8proc version information.

Definition at line 15 of file utf8.c.

References `utf8proc_version_d`.

Referenced by `lib_load()`.

5.374.1.3 `const chr_t* utf8_error_string (ssize_t error_code)`

Exchange an error code for a string explaining the nature of the UTF8 error.

Parameters:

error_code the numeric error returned by the internal UTF8 mapping logic.

Returns:

A string constant with the error message. If the code goes unrecognized, then a generic message is returned.

Definition at line 42 of file utf8.c.

References utf8proc_errmsg_d.

Referenced by utf8_length_st().

5.374.1.4 `size_t utf8_length_st (stringer_t * s)`

Determine how many valid Unicode characters the string has.

Parameters:

s the managed string to be tested.

Returns:

The number of codepoints found if the string is valid, otherwise 0.

Definition at line 51 of file utf8.c.

References bytes, log_pedantic, st_empty_out(), utf8_error_string(), utf8proc_get_property_d, and utf8proc_iterate_d.

Referenced by auth_login(), register_business_validate_password(), register_data_insert_user(), and stacie_derive_rounds().

5.374.1.5 `bool_t utf8_valid_st (stringer_t * s)`

Determine whether a managed string is a properly formatted UTF8 string.

Parameters:

s the managed string to be tested.

Returns:

Valid strings return true, while invalid strings return false.

Definition at line 90 of file utf8.c.

References bytes, log_pedantic, st_empty_out(), and utf8proc_iterate_d.

5.375 src/providers/parsers/xml.c File Reference

```
#include "magma.h"
```

Functions

- const [chr_t](#) * [lib_version_xml](#) (void)
Return the version string of libxml.
- [bool_t](#) [lib_load_xml](#) (void)
Initialize libxml and bind dynamically to the exported functions that are required.
- void [xml_node_free](#) (xmlNodePtr node)
Free an xml node.
- xmlChar * [xml_encode](#) (xmlDocPtr doc, [stringer_t](#) *string)
Encode an xml string, performing proper replacement of defined entities and non-ASCII values.
- [int_t](#) [xml_xpath_set_namespace](#) (xmlXPathContextPtr ctxt, xmlChar *prefix, xmlChar *ns_uri)
Set the namespace for an xpath context.
- xmlNodePtr [xml_node_add_sibling](#) (xmlNodePtr current, xmlNodePtr element)
Add an xml node to the list of siblings of another xml node.
- xmlNodePtr [xml_node_new](#) ([uchr_t](#) *name)
Create an xml node.
- [stringer_t](#) * [xml_node_get_content_st](#) (xmlNodePtr node)
Get the content of an xml node as a managed string.
- void [xml_node_set_content](#) (xmlNodePtr node, [uchr_t](#) *content)
Set the content of an xml node.
- xmlAttrPtr [xml_node_set_property](#) (xmlNodePtr node, [uchr_t](#) *name, [uchr_t](#) *value)
Set an attribute of an xml node.
- [uint64_t](#) [xml_get_xpath_uint64](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as an unsigned 64-bit integer.
- [uint32_t](#) [xml_get_xpath_uint32](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as an unsigned 32-bit integer.
- [uint16_t](#) [xml_get_xpath_uint16](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as an unsigned 16-bit integer.
- [uint8_t](#) [xml_get_xpath_uint8](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as an unsigned 8-bit integer.
- [int64_t](#) [xml_get_xpath_int64](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a signed 64-bit integer.
- [int32_t](#) [xml_get_xpath_int32](#) (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)

Get the value of a node element or attribute selected by an xpath expression as a signed 32-bit integer.

- `int16_t xml_get_xpath_int16` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a signed 16-bit integer.
- `int8_t xml_get_xpath_int8` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a signed 8-bit integer.
- `stringer_t * xml_get_xpath_st` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the value of a node element or attribute selected by an xpath expression as a managed string.
- `bool_t xml_set_xpath_uint64` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query, uint64_t val)
Set the value of a node element selected by an xpath expression, as a 64-bit unsigned integer.
- `bool_t xml_set_xpath_ns` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query, `uchr_t` *val)
Set the value of a node element selected by an xpath expression, as a null-terminated string.
- `bool_t xml_set_xpath_property` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query, `uchr_t` *name, `uchr_t` *val)
Set the value of a specified property of a node element selected by an xpath expression.
- `chr_t * xml_get_xpath_ns` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
Get the content of an evaluated xpath expression as a null-terminated string.
- `size_t xml_get_xpath_node_count` (xmlXPathContextPtr xpath_ctx, xmlChar *xpath_query)
This function is not currently being called by any other part of the code.
- `void xml_error` (void *ctx, const `chr_t` *format,...)
A function handler for displaying xml parser error messages to the user.
- `void xml_free_xpath_ctx` (xmlXPathContextPtr ctx)
Free an xml xpath context.
- `void xml_free_xpath_obj` (xmlXPathObjectPtr obj)
Free an xml xpath object.
- `xmlXPathContextPtr xml_create_xpath_ctx` (xmlDocPtr doc)
Create an xpath context for an xml document.
- `xmlXPathObjectPtr xml_xpath_eval` (const `uchr_t` *xpath, xmlXPathContextPtr ctx)
Evaluate an xml xpath expression.
- `xmlParserCtxtPtr xml_create_parser_ctx` (void)
Create and initialize a new xml parser context.
- `void xml_free_parser_ctx` (xmlParserCtxtPtr ctx)
Destroy an xml parser context.
- `void xml_free_doc` (xmlDocPtr doc)
Free an xml document.
- `stringer_t * xml_dump_doc` (xmlDocPtr doc)
Get an xml document object as a string.

- xmlDocPtr [xml_create_doc](#) (xmlParserCtxtPtr ctx, const [chr_t](#) *buffer, [int_t](#) size, const [chr_t](#) *url, const [chr_t](#) *encoding, [int_t](#) options)
Create a new xml document object from specified user data.
- void [xml_stop](#) (void)
Cleanup the state and allocated memory of the xml parser in preparation to be shutdown.
- [bool_t](#) [xml_start](#) (void)
Initialize the xml parser library.

Variables

- [chr_t](#) [xml_version_string](#) [9]
- pthread_mutex_t [log_mutex](#)

5.375.1 Function Documentation

5.375.1.1 bool_t lib_load_xml (void)

Initialize libxml and bind dynamically to the exported functions that are required. [xml.c](#)

Returns:

true on success or false on failure.

Definition at line 26 of file [xml.c](#).

References [lib_symbols\(\)](#), [M_BIND](#), [symbol_t](#), [uint64_conv_ns\(\)](#), [xml_version_string](#), and [xmlParserVersion_d](#).

Referenced by [lib_load\(\)](#).

5.375.1.2 const chr_t* lib_version_xml (void)

Return the version string of libxml.

Returns:

a pointer to a character string containing the libxml version information.

Definition at line 17 of file [xml.c](#).

References [xml_version_string](#).

Referenced by [lib_load\(\)](#).

5.375.1.3 xmlDocPtr xml_create_doc (xmlParserCtxtPtr ctx, const chr_t * buffer, int_t size, const chr_t * url, const chr_t * encoding, int_t options)

Create a new xml document object from specified user data.

See also:

[xmlCtxtReadMemory\(\)](#)

Parameters:

ctx a pointer to the xml context to be used for the parsing operation.

buffer a null-terminated string containing the xml data to be parsed.

size the length, in bytes, of the xml data buffer to be parsed.

url a null-terminated string containing the base url to be used for the document.

encoding a null-terminated string specifying the document encoding type, or NULL for default.

options a value containing a mask of xml parser options of type xmlParserOption.

Returns:

NULL on failure, or a pointer to the xml document tree of the parsed xml text on success.

Definition at line 1051 of file xml.c.

References log_pedantic, and xmlCtxtReadMemory_d.

Referenced by http_page_get().

5.375.1.4 xmlParserCtxtPtr xml_create_parser_ctx (void)

Create and initialize a new xml parser context.

Returns:

NULL on failure, or a newly initialized xml parser context on success.

Definition at line 961 of file xml.c.

References log_pedantic, xml_error(), and xmlNewParserCtxt_d.

Referenced by http_page_get().

5.375.1.5 xmlXPathContextPtr xml_create_xpath_ctx (xmlDocPtr doc)

Create an xpath context for an xml document.

Parameters:

doc a pointer to the input xml document object.

Returns:

NULL on failure, or a pointer to the new xpath context on success.

Definition at line 917 of file xml.c.

References log_pedantic, and xmlXPathNewContext_d.

Referenced by http_page_get().

5.375.1.6 stringer_t* xml_dump_doc (xmlDocPtr doc)

Get an xml document object as a string.

Parameters:

doc a pointer to the xml document object to be serialized.

Returns:

NULL on failure or a pointer to a managed string containing the specified xml document's serialized data on success.

Definition at line 1018 of file xml.c.

References mm_free(), st_import(), and xmlDocDumpFormatMemory_d.

Referenced by contact_print_form(), contact_print_message(), portal_print_login(), statistics_process(), teacher_print_form(), and teacher_print_message().

5.375.1.7 xmlChar* xml_encode (xmlDocPtr *doc*, stringer_t * *string*)

Encode an xml string, performing proper replacement of defined entities and non-ASCII values.

Parameters:

doc a pointer to the xml document containing the DTD to be used for the encoding process.

string a managed string containing the xml data to be encoded.

NULL on failure, or a newly allocated null-terminated string containing the encoded xml data on success.

Definition at line 69 of file xml.c.

References st_data_get(), and xmlEncodeEntitiesReentrant_d.

Referenced by contact_print_form().

5.375.1.8 void xml_error (void * *ctx*, const chr_t * *format*, ...)

A function handler for displaying xml parser error messages to the user.

Parameters:

ctx a placeholder; ignored.

format a null-terminated string containing a format string for the error message to be displayed.

... a variable argument list containing the parameters to the format string.

Returns:

This function returns no value.

Definition at line 854 of file xml.c.

References log_internal(), M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_LINE_DISABLE, M_LOG_LINE_FEED_DISABLE, M_LOG_STACK_TRACE_DISABLE, and M_LOG_TIME_DISABLE.

Referenced by xml_create_parser_ctx().

5.375.1.9 void xml_free_doc (xmlDocPtr *doc*)

Free an xml document.

Parameters:

doc a pointer to the xml document to be freed.

Returns:

This function returns no value.

Definition at line 1001 of file xml.c.

References log_pedantic, and xmlFreeDoc_d.

Referenced by http_page_free().

5.375.1.10 void xml_free_parser_ctx (xmlParserCtxtPtr *ctx*)

Destroy an xml parser context.

Parameters:

ctx a pointer to the xml parser context to be freed.

Returns:

This function returns no value.

Definition at line 984 of file xml.c.

References log_pedantic, and xmlFreeParserCtxt_d.

Referenced by http_page_free().

5.375.1.11 void xml_free_xpath_ctx (xmlXPathContextPtr *ctx*)

Free an xml xpath context.

Parameters:

ctx a pointer to the xml xpath context to be freed.

Returns:

This function returns no value.

Definition at line 883 of file xml.c.

References log_pedantic, and xmlXPathFreeContext_d.

Referenced by http_page_free().

5.375.1.12 void xml_free_xpath_obj (xmlXPathObjectPtr *obj*)

Free an xml xpath object.

Parameters:

obj a pointer to the xml xpath object to be freed.

Returns:

This function returns no value.

Definition at line 900 of file xml.c.

References log_pedantic, and xmlXPathFreeObject_d.

Referenced by xml_set_xpath_ns(), xml_set_xpath_property(), and xml_set_xpath_uint64().

5.375.1.13 int16_t xml_get_xpath_int16 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as a signed 16-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a signed 16-bit integer.

Definition at line 503 of file xml.c.

References `int16_conv_st()`, `log_pedantic`, `ns_length_get()`, `PLACER`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.375.1.14 int32_t xml_get_xpath_int32 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as a signed 32-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a signed 32-bit integer.

Definition at line 448 of file xml.c.

References `int32_conv_st()`, `log_pedantic`, `ns_length_get()`, `PLACER`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.375.1.15 int64_t xml_get_xpath_int64 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as a signed 64-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a signed 64-bit integer.

Definition at line 393 of file xml.c.

References `int64_conv_st()`, `log_pedantic`, `ns_length_get()`, `PLACER`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.375.1.16 int8_t xml_get_xpath_int8 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as a signed 8-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a signed 8-bit integer.

Definition at line 558 of file xml.c.

References `int8_conv_st()`, `log_pedantic`, `ns_length_get()`, `PLACER`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.375.1.17 `size_t xml_get_xpath_node_count (xmlXPathContextPtr xpath_ctx, xmlChar * xpath_query)`

This function is not currently being called by any other part of the code.

Definition at line 817 of file xml.c.

References `log_pedantic`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.375.1.18 `chr_t* xml_get_xpath_ns (xmlXPathContextPtr xpath_ctx, xmlChar * xpath_query)`

Get the content of an evaluated xpath expression as a null-terminated string.

Parameters:

ctx a pointer to the xpath context to be used for the evaluation.

xpath_query a null-terminated string containing the xpath expression to be evaluated.

Returns:

NULL on failure, or a null-terminated string containing the value of the xpath expression's specified node on success.

Definition at line 766 of file xml.c.

References `log_pedantic`, `ns_dupe()`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.375.1.19 `stringer_t* xml_get_xpath_st (xmlXPathContextPtr xpath_ctx, xmlChar * xpath_query)`

Get the value of a node element or attribute selected by an xpath expression as a managed string.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as a managed string.

Definition at line 613 of file xml.c.

References `log_pedantic`, `ns_length_get()`, `st_import()`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.375.1.20 `uint16_t xml_get_xpath_uint16 (xmlXPathContextPtr xpath_ctx, xmlChar * xpath_query)`

Get the value of a node element or attribute selected by an xpath expression as an unsigned 16-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as an unsigned 16-bit integer.

Definition at line 283 of file xml.c.

References `log_pedantic`, `ns_length_get()`, `PLACER`, `uint16_conv_st()`, `xmlXPathEvalExpression_d`, and `xmlXPathFreeObject_d`.

5.375.1.21 uint32_t xml_get_xpath_uint32 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as an unsigned 32-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as an unsigned 32-bit integer.

Definition at line 229 of file xml.c.

References log_pedantic, ns_length_get(), PLACER, uint32_conv_st(), xmlXPathEvalExpression_d, and xmlXPathFreeObject_d.

5.375.1.22 uint64_t xml_get_xpath_uint64 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as an unsigned 64-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as an unsigned 64-bit integer.

Definition at line 174 of file xml.c.

References log_pedantic, ns_length_get(), PLACER, uint64_conv_st(), xmlXPathEvalExpression_d, and xmlXPathFreeObject_d.

5.375.1.23 uint8_t xml_get_xpath_uint8 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*)

Get the value of a node element or attribute selected by an xpath expression as an unsigned 8-bit integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

Returns:

0 on failure, or the value of the selected node as an unsigned 8-bit integer.

Definition at line 338 of file xml.c.

References log_pedantic, ns_length_get(), PLACER, uint8_conv_st(), xmlXPathEvalExpression_d, and xmlXPathFreeObject_d.

5.375.1.24 xmlNodePtr xml_node_add_sibling (xmlNodePtr *current*, xmlNodePtr *element*)

Add an xml node to the list of siblings of another xml node.

Note:

If the sibling node was already inserted into a document it will first be unlinked.

Parameters:

current a pointer to the xml node to which the sibling node will be added.

element a pointer to the sibling node to be added to the target node.

Returns:

NULL on failure, or a pointer to the sibling node on success.

Definition at line 94 of file xml.c.

References xmlAddSibling_d.

Referenced by contact_business_add_error(), and teacher_add_error().

5.375.1.25 void xml_node_free (xmlNodePtr node)

Free an xml node.

Parameters:

node a pointer to the xml node to be freed.

Returns:

This function returns no value.

Definition at line 57 of file xml.c.

References xmlFreeNode_d.

Referenced by contact_business_add_error(), and teacher_add_error().

5.375.1.26 stringer_t* xml_node_get_content_st (xmlNodePtr node)

Get the content of an xml node as a managed string.

Parameters:

node a pointer to the xml node to be queried.

Returns:

NULL on failure, or a pointer to a managed string

Definition at line 114 of file xml.c.

References log_pedantic, st_import(), xmlBufferContent_d, xmlBufferCreate_d, xmlBufferFree_d, xmlBufferLength_d, and xmlNodeBufGetContent_d.

5.375.1.27 xmlNodePtr xml_node_new (uchar_t * name)

Create an xml node.

Parameters:

name the name of the new xml node.

Returns:

NULL on failure, or a pointer to the newly allocated and initialized xml node on success.

Definition at line 104 of file xml.c.

References xmlNewNode_d.

Referenced by contact_business_add_error(), and teacher_add_error().

5.375.1.28 void xml_node_set_content (xmlNodePtr *node*, uchr_t * *content*)

Set the content of an xml node.

Parameters:

node a pointer to the xml node to be set.

content a null-terminated string containing the new content of the xml node.

Returns:

This function returns no value.

Definition at line 150 of file xml.c.

References xmlNodeSetContent_d.

Referenced by contact_business_add_error(), portal_print_login(), teacher_add_error(), xml_set_xpath_ns(), and xml_set_xpath_uint64().

5.375.1.29 xmlAttrPtr xml_node_set_property (xmlNodePtr *node*, uchr_t * *name*, uchr_t * *value*)

Set an attribute of an xml node.

Parameters:

node a pointer to the xml node to be set.

name a null-terminated string containing the name of the node attribute to be set.

value a null-terminated string containing the new value of the specified xml node's attribute.

Returns:

NULL on failure, or a pointer to node's attribute on success.

Definition at line 163 of file xml.c.

References xmlSetProp_d.

Referenced by contact_business_add_error(), teacher_add_error(), and xml_set_xpath_property().

5.375.1.30 bool_t xml_set_xpath_ns (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*, uchr_t * *val*)

Set the value of a node element selected by an xpath expression, as a null-terminated string.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

val the new value of the selected node, as a null-terminated string.

Returns:

true if the value was set successfully, or false on failure.

Definition at line 702 of file xml.c.

References log_pedantic, xml_free_xpath_obj(), xml_node_set_content(), and xml_xpath_eval().

Referenced by contact_print_form(), contact_print_message(), statistics_process(), teacher_print_form(), and teacher_print_message().

5.375.1.31 bool_t xml_set_xpath_property (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*, uchr_t * *name*, uchr_t * *val*)

Set the value of a specified property of a node element selected by an xpath expression.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

name a null-terminated string containing the name of the node's property to be set.

val the new value of the selected node's property, as a null-terminated string.

Returns:

true if the selected node's value was set successfully, or false on failure.

Definition at line 735 of file xml.c.

References log_pedantic, xml_free_xpath_obj(), xml_node_set_property(), and xml_xpath_eval().

Referenced by contact_print_form(), contact_print_message(), and teacher_print_form().

5.375.1.32 bool_t xml_set_xpath_uint64 (xmlXPathContextPtr *xpath_ctx*, xmlChar * *xpath_query*, uint64_t *val*)

Set the value of a node element selected by an xpath expression, as a 64-bit unsigned integer.

Parameters:

xpath_ctx a pointer to the specified xpath context.

xpath_query a null-terminated string containing the xpath expression to select the desired node.

val the new value of the selected node, as a 64-bit unsigned integer.

Returns:

true if the value was set successfully, or false on failure.

Definition at line 668 of file xml.c.

References log_pedantic, xml_free_xpath_obj(), xml_node_set_content(), and xml_xpath_eval().

Referenced by statistics_process().

5.375.1.33 bool_t xml_start (void)

Initialize the xml parser library.

Returns:

This function returns no value.

Definition at line 1079 of file xml.c.

References xmlInitParser_d.

Referenced by process_start().

5.375.1.34 void xml_stop (void)

Cleanup the state and allocated memory of the xml parser in preparation to be shutdown.

Returns:

This function returns no value.

Definition at line 1066 of file xml.c.

References xmlCleanupGlobals_d, xmlCleanupParser_d, and xmlMemoryDump_d.

Referenced by process_stop().

5.375.1.35 xmlXPathObjectPtr xml_xpath_eval (const uchr_t * *xpath*, xmlXPathContextPtr *ctx*)

Evaluate an xml xpath expression.

Parameters:

a null-terminated string containing the xpath expression to be evaluated.

a pointer to the xpath context in which to perform the evaluation. NULL on failure, or a pointer to the xml path object of the evaluation on success.

Definition at line 941 of file xml.c.

References log_pedantic, and xmlXPathEvalExpression_d.

Referenced by contact_business_add_error(), portal_print_login(), teacher_add_error(), xml_set_xpath_ns(), xml_set_xpath_property(), and xml_set_xpath_uint64().

5.375.1.36 int_t xml_xpath_set_namespace (xmlXPathContextPtr *ctxt*, xmlChar * *prefix*, xmlChar * *ns_uri*)

Set the namespace for an xpath context.

See also:

xmlXPathRegisterNs()

Parameters:

ctxt a pointer to the specified xpath context.

prefix a pointer to the prefix of the namespace as a string.

ns_uri a pointer to the uri of the namespace as a string.

Returns:

-1 on failure or 0 on success.

Definition at line 82 of file xml.c.

References xmlXPathRegisterNs_d.

Referenced by http_page_get().

5.375.2 Variable Documentation**5.375.2.1 pthread_mutex_t log_mutex**

Definition at line 13 of file log.c.

5.375.2.2 chr_t xml_version_string[9]

Definition at line 10 of file xml.c.

Referenced by lib_load_xml(), and lib_version_xml().

5.376 src/providers/prime/cryptography/aes.c File Reference

```
#include "magma.h"
```

Defines

- `#define PRIME_CHUNK_PREFIX_LEN 4`
- `#define PRIME_CHUNK_KEY_LEN 64`
- `#define PRIME_CHUNK_HEAD_LEN 36`
- `#define PRIME_OBJECT_PREFIX_LEN 6`
- `#define PRIME_OBJECT_KEY_LEN 64`
- `#define PRIME_OBJECT_HEAD_LEN 38`
- `#define PRIME_OBJECT_PAYLOAD_PREFIX_LEN 4`

Functions

- `struct __attribute__((packed))`
- `placer_t aes_cipher_key (stringer_t *key)`
Extract the symmetric cipher portion of the key.
- `placer_t aes_vector_shard (stringer_t *key)`
Extract the portion of the key used for the vector shard.
- `placer_t aes_tag_shard (stringer_t *key)`
Extract the portion of the key used for the tag shard.
- `stringer_t * aes_chunk_encrypt (uint8_t type, stringer_t *key, stringer_t *chunk, stringer_t *output)`
Create an encrypted chunk. The result includes the 1 byte type, the 3 byte big endian length, the 16 byte tag and the 16 byte vector, followed by the encrypted data. Note the caller must pad the data buffer to a 16 byte boundary.
- `stringer_t * aes_chunk_decrypt (stringer_t *key, stringer_t *chunk, stringer_t *output)`
- `stringer_t * aes_artifact_encrypt (stringer_t *key, stringer_t *object, stringer_t *output)`
- `stringer_t * aes_artifact_decrypt (stringer_t *key, stringer_t *object, stringer_t *output)`

Variables

- `prime_encrypted_object_header_t`
- `prime_encrypted_chunk_header_t`
- `prime_encrypted_payload_prefix_t`

5.376.1 Define Documentation

5.376.1.1 `#define PRIME_CHUNK_HEAD_LEN 36`

Definition at line 12 of file aes.c.

Referenced by `aes_chunk_decrypt()`, and `aes_chunk_encrypt()`.

5.376.1.2 `#define PRIME_CHUNK_KEY_LEN 64`

Definition at line 11 of file aes.c.

Referenced by `aes_chunk_decrypt()`, and `aes_chunk_encrypt()`.

5.376.1.3 #define PRIME_CHUNK_PREFIX_LEN 4

Definition at line 10 of file aes.c.

Referenced by aes_chunk_decrypt().

5.376.1.4 #define PRIME_OBJECT_HEAD_LEN 38

Definition at line 16 of file aes.c.

Referenced by aes_artifact_decrypt(), and aes_artifact_encrypt().

5.376.1.5 #define PRIME_OBJECT_KEY_LEN 64

Definition at line 15 of file aes.c.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_cipher_key(), aes_tag_shard(), and aes_vector_shard().

5.376.1.6 #define PRIME_OBJECT_PAYLOAD_PREFIX_LEN 4

Definition at line 17 of file aes.c.

Referenced by aes_artifact_decrypt(), and aes_artifact_encrypt().

5.376.1.7 #define PRIME_OBJECT_PREFIX_LEN 6

Definition at line 14 of file aes.c.

Referenced by aes_artifact_decrypt().

5.376.2 Function Documentation**5.376.2.1 struct __attribute__((packed)) [read]**

Definition at line 42 of file aes.c.

5.376.2.2 stringer_t* aes_artifact_decrypt (stringer_t * key, stringer_t * object, stringer_t * output)

Definition at line 627 of file aes.c.

References AES_BLOCK_LEN, aes_cipher_key(), AES_KEY_LEN, AES_TAG_LEN, aes_tag_shard(), AES_VECTOR_LEN, aes_vector_shard(), EVP_aes_256_gcm_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_key_length_d, EVP_DecryptFinal_ex_d, EVP_DecryptInit_ex_d, EVP_DecryptUpdate_d, log_pedantic, MANAGEDBUF, MEMORYBUF, mm_copy(), mm_move(), pl_data_get(), pl_empty(), pl_null(), PLACER, placer_t, prime_encrypted_object_header_t, prime_header_write(), PRIME_OBJECT_HEAD_LEN, PRIME_OBJECT_KEY_LEN, PRIME_OBJECT_PAYLOAD_PREFIX_LEN, PRIME_OBJECT_PREFIX_LEN, PRIME_ORG_KEY, PRIME_ORG_KEY_ENCRYPTED, PRIME_USER_KEY, PRIME_USER_KEY_ENCRYPTED, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), st_wipe(), st_xor(), and type().

Referenced by org_encrypted_key_set(), and user_encrypted_key_set().

5.376.2.3 stringer_t* aes_artifact_encrypt (stringer_t * key, stringer_t * object, stringer_t * output)

Definition at line 417 of file aes.c.

References AES_BLOCK_LEN, aes_cipher_key(), AES_KEY_LEN, AES_TAG_LEN, aes_tag_shard(), AES_VECTOR_LEN, aes_vector_shard(), EVP_aes_256_gcm_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_key_length_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, log_pedantic, MANAGEDBUF, MEMORYBUF, mm_copy(), pl_data_get(), pl_empty(), pl_null(), PLACER, placer_t, prime_encrypted_object_header_t, prime_header_read(), PRIME_OBJECT_HEAD_LEN, PRIME_OBJECT_KEY_LEN, PRIME_OBJECT_PAYLOAD_PREFIX_LEN, PRIME_ORG_KEY, PRIME_ORG_KEY_ENCRYPTED, PRIME_USER_KEY, PRIME_USER_KEY_ENCRYPTED, rand_write(), ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), st_wipe(), st_xor(), and type().

Referenced by org_encrypted_key_get(), and user_encrypted_key_get().

5.376.2.4 stringer_t* aes_chunk_decrypt (stringer_t * key, stringer_t * chunk, stringer_t * output)

Definition at line 275 of file aes.c.

References aes_cipher_key(), AES_KEY_LEN, AES_TAG_LEN, aes_tag_shard(), AES_VECTOR_LEN, aes_vector_shard(), EVP_aes_256_gcm_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_key_length_d, EVP_DecryptFinal_ex_d, EVP_DecryptInit_ex_d, EVP_DecryptUpdate_d, log_pedantic, MANAGEDBUF, MEMORYBUF, mm_copy(), pl_data_get(), pl_empty(), pl_null(), PLACER, placer_t, PRIME_CHUNK_HEAD_LEN, PRIME_CHUNK_KEY_LEN, PRIME_CHUNK_PREFIX_LEN, prime_encrypted_chunk_header_t, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), st_wipe(), and st_xor().

Referenced by encrypted_chunk_get().

5.376.2.5 stringer_t* aes_chunk_encrypt (uint8_t type, stringer_t * key, stringer_t * chunk, stringer_t * output)

Create an encrypted chunk. The result includes the 1 byte type, the 3 byte big endian length, the 16 byte tag and the 16 byte vector, followed by the encrypted data. Note the caller must pad the data buffer to a 16 byte boundary.

Definition at line 118 of file aes.c.

References AES_BLOCK_LEN, aes_cipher_key(), AES_KEY_LEN, AES_TAG_LEN, aes_tag_shard(), AES_VECTOR_LEN, aes_vector_shard(), EVP_aes_256_gcm_d, EVP_CIPHER_CTX_cleanup_d, EVP_CIPHER_CTX_ctrl_d, EVP_CIPHER_CTX_init_d, EVP_CIPHER_CTX_key_length_d, EVP_EncryptFinal_ex_d, EVP_EncryptInit_ex_d, EVP_EncryptUpdate_d, log_pedantic, MANAGEDBUF, MEMORYBUF, mm_copy(), pl_data_get(), pl_empty(), pl_null(), PLACER, placer_t, PRIME_CHUNK_HEAD_LEN, PRIME_CHUNK_KEY_LEN, prime_encrypted_chunk_header_t, rand_write(), ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), st_wipe(), and st_xor().

Referenced by encrypted_chunk_set().

5.376.2.6 placer_t aes_cipher_key (stringer_t * key)

Extract the symmetric cipher portion of the key. [aes.c](#)

Parameters:

key The complete key, which holds the vector and tag shard values along with the cipher key.

Returns:

returns a place holder with the cipher key portion, or NULL if the supplied key was invalid.

Definition at line 58 of file aes.c.

References AES_KEY_LEN, AES_TAG_LEN, AES_VECTOR_LEN, log_error, pl_init(), pl_null(), placer_t, PRIME_OBJECT_KEY_LEN, st_data_get(), st_empty, and st_length_get().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), and aes_chunk_encrypt().

5.376.2.7 `placer_t aes_tag_shard (stringer_t * key)`

Extract the portion of the key used for the tag shard.

Parameters:

key The complete key, which holds the vector and tag shard values along with the cipher key.

Returns:

returns a placeholder with the tag shard, or NULL if the supplied key was invalid.

Definition at line 100 of file aes.c.

References AES_TAG_LEN, AES_VECTOR_LEN, log_error, pl_init(), pl_null(), placer_t, PRIME_OBJECT_KEY_LEN, st_data_get(), st_empty, and st_length_get().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), and aes_chunk_encrypt().

5.376.2.8 `placer_t aes_vector_shard (stringer_t * key)`

Extract the portion of the key used for the vector shard.

Parameters:

key The complete key, which holds the vector and tag shard values along with the cipher key.

Returns:

returns a place holder with the vector shard, or NULL if the supplied key was invalid.

Definition at line 79 of file aes.c.

References AES_VECTOR_LEN, log_error, pl_init(), pl_null(), placer_t, PRIME_OBJECT_KEY_LEN, st_data_get(), st_empty, and st_length_get().

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), and aes_chunk_encrypt().

5.376.3 Variable Documentation

5.376.3.1 `prime_encrypted_chunk_header_t`

Definition at line 39 of file aes.c.

Referenced by aes_chunk_decrypt(), and aes_chunk_encrypt().

5.376.3.2 `prime_encrypted_object_header_t`

Definition at line 28 of file aes.c.

Referenced by aes_artifact_decrypt(), and aes_artifact_encrypt().

5.376.3.3 `prime_encrypted_payload_prefix_t`

Definition at line 49 of file aes.c.

5.377 src/providers/prime/cryptography/secp256k1.c File Reference

```
#include "magma.h"
```

Functions

- [secp256k1_key_type_t secp256k1_type](#) ([secp256k1_key_t](#) *key)
- [void secp256k1_free](#) ([secp256k1_key_t](#) *key)
Free a secp256k1 key structure.
- [secp256k1_key_t * secp256k1_alloc](#) (void)
Allocate a new key pair using the secp256k1 curve.
- [secp256k1_key_t * secp256k1_generate](#) (void)
Generate a random secp256k1 key pair.
- [stringer_t * secp256k1_public_get](#) ([secp256k1_key_t](#) *key, [stringer_t](#) *output)
Return a secp256k1 public key as a compressed point encoded as a big endian integer.
- [stringer_t * secp256k1_private_get](#) ([secp256k1_key_t](#) *key, [stringer_t](#) *output)
Return a secp256k1 private key as a big endian integer inside a managed string.
- [secp256k1_key_t * secp256k1_public_set](#) ([stringer_t](#) *key)
- [secp256k1_key_t * secp256k1_private_set](#) ([stringer_t](#) *key)
- [stringer_t * secp256k1_compute_kek](#) ([secp256k1_key_t](#) *priv, [secp256k1_key_t](#) *pub, [stringer_t](#) *output)

Variables

- EC_GROUP * [prime_curve_group](#)

5.377.1 Function Documentation

5.377.1.1 [secp256k1_key_t* secp256k1_alloc](#) (void)

Allocate a new key pair using the secp256k1 curve. [secp256k1.c](#)

See also:

[NID_secp256k1](#)

Returns:

NULL on failure, or a pointer to the newly allocated key pair.

Definition at line 53 of file [secp256k1.c](#).

References [EC_KEY_free_d](#), [EC_KEY_new_by_curve_name_d](#), [EC_KEY_new_d](#), [EC_KEY_set_conv_form_d](#), [EC_KEY_set_group_d](#), [log_info](#), [MEMORYBUF](#), [prime_curve_group](#), and [ssl_error_string\(\)](#).

Referenced by [secp256k1_generate\(\)](#), [secp256k1_private_set\(\)](#), and [secp256k1_public_set\(\)](#).

5.377.1.2 stringer_t* secp256k1_compute_kek (secp256k1_key_t * *priv*, secp256k1_key_t * *pub*, stringer_t * *output*)

Parameters:

private

public

output

Returns:

Definition at line 321 of file secp256k1.c.

References EC_KEY_get0_public_key_d, ECDH_compute_key_d, log_info, log_pedantic, MEMORYBUF, SECP256K1_SHARED_SECRET_LEN, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), and st_valid_tracked().

Referenced by keks_get(), and keks_set().

5.377.1.3 void secp256k1_free (secp256k1_key_t * *key*)

Free a secp256k1 key structure.

See also:

EC_KEY_free()

Parameters:

key the managed string to be freed.

Returns:

This function returns no value.

Definition at line 33 of file secp256k1.c.

References EC_KEY_free_d, and log_pedantic.

Referenced by encrypted_message_free(), ephemeral_chunk_free(), org_key_free(), org_signet_free(), user_key_free(), and user_signet_free().

5.377.1.4 secp256k1_key_t* secp256k1_generate (void)

Generate a random secp256k1 key pair.

Returns:

NULL on failure, or a new, randomly generated secp256k1 key pair on success.

Definition at line 86 of file secp256k1.c.

References EC_KEY_free_d, EC_KEY_generate_key_d, log_info, MEMORYBUF, secp256k1_alloc(), and ssl_error_string().

Referenced by _generate_ec_keypair(), naked_message_set(), org_key_generate(), and user_key_generate().

5.377.1.5 stringer_t* secp256k1_private_get (secp256k1_key_t * *key*, stringer_t * *output*)

Return a secp256k1 private key as a big endian integer inside a managed string.

Parameters:

key the input secp256k1 key pair.

Returns:

NULL on failure, or the private key as a big endian integer.

Definition at line 151 of file secp256k1.c.

References BN_bn2bin_d, BN_num_bits_d, BN_num_bytes_d, EC_KEY_get0_private_key_d, log_pedantic, MEMORYBUF, SECP256K1_KEY_PRIV_LEN, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and st_wipe().

Referenced by _serialize_ec_privkey(), org_key_get(), and user_key_get().

5.377.1.6 secp256k1_key_t* secp256k1_private_set (stringer_t * *key*)**Parameters:**

key

Returns:

Definition at line 257 of file secp256k1.c.

References BN_bin2bn_d, BN_CTX_free_d, BN_CTX_new_d, BN_CTX_start_d, BN_free_d, EC_KEY_check_key_d, EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_set_private_key_d, EC_KEY_set_public_key_d, EC_POINT_free_d, EC_POINT_mul_d, EC_POINT_new_d, log_info, MEMORYBUF, number, secp256k1_alloc(), SECP256K1_KEY_PRIV_LEN, ssl_error_string(), st_data_get(), st_empty, and st_length_get().

Referenced by _deserialize_ec_privkey(), _load_ec_privkey(), org_key_set(), and user_key_set().

5.377.1.7 stringer_t* secp256k1_public_get (secp256k1_key_t * *key*, stringer_t * *output*)

Return a secp256k1 public key as a compressed point encoded as a big endian integer.

Parameters:

key the input secp256k1 key pair.

Returns:

NULL on failure, or the public key as a big endian integer.

Definition at line 110 of file secp256k1.c.

References EC_KEY_get0_group_d, EC_KEY_get0_public_key_d, EC_KEY_get_conv_form_d, EC_POINT_point2oct_d, log_pedantic, MEMORYBUF, SECP256K1_KEY_PUB_LEN, ssl_error_string(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), and st_valid_tracked().

Referenced by _serialize_ec_pubkey(), ephemeral_chunk_get(), naked_message_set(), org_signet_fingerprint(), org_signet_generate(), org_signet_get(), org_signet_verify(), user_request_generate(), user_request_get(), user_request_rotation(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_get(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.377.1.8 secp256k1_key_t* secp256k1_public_set (stringer_t * *key*)**Parameters:**

key

Returns:

Definition at line 205 of file secp256k1.c.

References BN_CTX_free_d, BN_CTX_new_d, BN_CTX_start_d, EC_KEY_free_d, EC_KEY_get0_group_d, EC_KEY_set_public_key_d, EC_POINT_free_d, EC_POINT_new_d, EC_POINT_oct2point_d, log_info, MEMORYBUF, secp256k1_alloc(), SECP256K1_KEY_PUB_LEN, ssl_error_string(), st_char_get(), st_data_get(), st_empty, and st_length_get().

Referenced by ephemeral_chunk_get(), ephemeral_chunk_set(), naked_message_set(), org_signet_generate(), org_signet_set(), user_request_generate(), user_request_rotation(), user_request_set(), user_request_sign(), and user_signet_set().

5.377.1.9 secp256k1_key_type_t secp256k1_type (secp256k1_key_t * *key*)

Definition at line 12 of file secp256k1.c.

References EC_KEY_get0_private_key_d, EC_KEY_get0_public_key_d, SECP256K1_ERR, SECP256K1_PRIV, and SECP256K1_PUB.

Referenced by keks_get(), and keks_set().

5.377.2 Variable Documentation**5.377.2.1 EC_GROUP* prime_curve_group**

Definition at line 12 of file prime.c.

Referenced by prime_start(), prime_stop(), and secp256k1_alloc().

5.378 src/providers/prime/keys/orgs.c File Reference

```
#include "magma.h"
```

Functions

- void [org_key_free](#) ([prime_org_key_t](#) *org)
- [prime_org_key_t](#) * [org_key_alloc](#) (void)
- [prime_org_key_t](#) * [org_key_generate](#) (void)
- size_t [org_key_length](#) ([prime_org_key_t](#) *org)
- [stringer_t](#) * [org_key_get](#) ([prime_org_key_t](#) *org, [stringer_t](#) *output)
- [prime_org_key_t](#) * [org_key_set](#) ([stringer_t](#) *org)
- [stringer_t](#) * [org_encrypted_key_get](#) ([stringer_t](#) *key, [prime_org_key_t](#) *org, [stringer_t](#) *output)

orgs.c

- [prime_org_key_t](#) * [org_encrypted_key_set](#) ([stringer_t](#) *key, [stringer_t](#) *org)

5.378.1 Function Documentation

5.378.1.1 [stringer_t](#)* [org_encrypted_key_get](#) ([stringer_t](#) *key, [prime_org_key_t](#) *org, [stringer_t](#) *output)

orgs.c

Definition at line 142 of file *orgs.c*.

References [aes_artifact_encrypt\(\)](#), [org_key_get\(\)](#), and [st_free\(\)](#).

Referenced by [prime_key_encrypt\(\)](#).

5.378.1.2 [prime_org_key_t](#)* [org_encrypted_key_set](#) ([stringer_t](#) *key, [stringer_t](#) *org)

Definition at line 159 of file *orgs.c*.

References [aes_artifact_decrypt\(\)](#), [org_key_set\(\)](#), [prime_org_key_t](#), and [st_free\(\)](#).

Referenced by [prime_key_decrypt\(\)](#).

5.378.1.3 [prime_org_key_t](#)* [org_key_alloc](#) (void)

Definition at line 21 of file *orgs.c*.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_org_key_t](#).

Referenced by [org_key_set\(\)](#).

5.378.1.4 void [org_key_free](#) ([prime_org_key_t](#) *org)

Definition at line 10 of file *orgs.c*.

References [ed25519_free\(\)](#), [mm_free\(\)](#), and [secp256k1_free\(\)](#).

Referenced by [org_key_generate\(\)](#), [org_key_set\(\)](#), and [prime_free\(\)](#).

5.378.1.5 prime_org_key_t* org_key_generate (void)

Definition at line 35 of file orgs.c.

References ed25519_generate(), log_pedantic, mm_alloc(), org_key_free(), prime_org_key_t, and secp256k1_generate().

Referenced by prime_key_generate().

5.378.1.6 stringer_t* org_key_get (prime_org_key_t * org, stringer_t * output)

Definition at line 61 of file orgs.c.

References ED25519_KEY_PRIV_LEN, ed25519_private_get(), length, log_pedantic, MANAGEDBUF, org_key_length(), prime_field_write(), prime_header_org_key_write(), PRIME_ORG_KEY, SECP256K1_KEY_PRIV_LEN, secp256k1_private_get(), st_alloc(), st_avail_get(), st_cleanup, st_opt_get(), st_valid_destination(), st_wipe(), and st_write.

Referenced by org_encrypted_key_get(), and prime_get().

5.378.1.7 size_t org_key_length (prime_org_key_t * org)

Definition at line 52 of file orgs.c.

Referenced by org_key_get().

5.378.1.8 prime_org_key_t* org_key_set (stringer_t * org)

Definition at line 100 of file orgs.c.

References ed25519_private_set(), log_pedantic, org_key_alloc(), org_key_free(), prime_field_get(), prime_field_t, prime_object_free(), prime_object_t, PRIME_ORG_KEY, prime_org_key_t, prime_unpack(), and secp256k1_private_set().

Referenced by org_encrypted_key_set(), and prime_set().

5.379 src/providers/prime/signets/orgs.c File Reference

```
#include "magma.h"
```

Functions

- void [org_signet_free](#) ([prime_org_signet_t](#) *org)
- [prime_org_signet_t](#) * [org_signet_alloc](#) (void)
orgs.c
- [prime_org_signet_t](#) * [org_signet_generate](#) ([prime_org_key_t](#) *org)
Derive an organizational signet from the corresponding private key structures.
- size_t [org_signet_length](#) ([prime_org_signet_t](#) *org)
- [stringer_t](#) * [org_signet_get](#) ([prime_org_signet_t](#) *org, [stringer_t](#) *output)
- [prime_org_signet_t](#) * [org_signet_set](#) ([stringer_t](#) *org)
- bool_t [org_signet_verify](#) ([prime_org_signet_t](#) *org)
- [stringer_t](#) * [org_signet_fingerprint](#) ([prime_org_signet_t](#) *org, [stringer_t](#) *output)

5.379.1 Function Documentation

5.379.1.1 [prime_org_signet_t](#)* [org_signet_alloc](#) (void)

orgs.c

Definition at line 24 of file *orgs.c*.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_org_signet_t](#).

Referenced by [org_signet_set\(\)](#).

5.379.1.2 [stringer_t](#)* [org_signet_fingerprint](#) ([prime_org_signet_t](#) * *org*, [stringer_t](#) * *output*)

Definition at line 230 of file *orgs.c*.

References [ED25519_KEY_PUB_LEN](#), [ed25519_public_get\(\)](#), [ED25519_SIGNATURE_LEN](#), [hash_sha512\(\)](#), [MANAGEDBUF](#), [prime_field_write\(\)](#), [PRIME_ORG_SIGNET](#), [SECP256K1_KEY_PUB_LEN](#), [secp256k1_public_get\(\)](#), [st_length_get\(\)](#), and [st_write](#).

Referenced by [prime_signet_fingerprint\(\)](#).

5.379.1.3 void [org_signet_free](#) ([prime_org_signet_t](#) * *org*)

Definition at line 12 of file *orgs.c*.

References [ed25519_free\(\)](#), [mm_free\(\)](#), [secp256k1_free\(\)](#), and [st_free\(\)](#).

Referenced by [org_signet_generate\(\)](#), [org_signet_set\(\)](#), and [prime_free\(\)](#).

5.379.1.4 [prime_org_signet_t](#)* [org_signet_generate](#) ([prime_org_key_t](#) * *org*)

Derive an organizational signet from the corresponding private key structures.

Definition at line 41 of file *orgs.c*.

References [ED25519_KEY_PUB_LEN](#), [ED25519_PRIV](#), [ed25519_public_get\(\)](#), [ed25519_public_set\(\)](#), [ed25519_sign\(\)](#), [log_pedantic](#), [MANAGEDBUF](#), [mm_alloc\(\)](#), [org_signet_free\(\)](#), [prime_field_write\(\)](#), [PRIME_ORG_SIGNET](#), [prime_org_signet_t](#), [SECP256K1_KEY_PUB_LEN](#), [secp256k1_public_get\(\)](#), [secp256k1_public_set\(\)](#), [st_cleanup](#), [st_free\(\)](#), and [st_write](#).

Referenced by `prime_signet_generate()`.

5.379.1.5 stringer_t* org_signet_get (prime_org_signet_t * org, stringer_t * output)

Definition at line 106 of file `orgs.c`.

References `ED25519_KEY_PUB_LEN`, `ed25519_public_get()`, `ED25519_SIGNATURE_LEN`, `length`, `log_pedantic`, `MANAGEDBUF`, `org_signet_length()`, `prime_field_write()`, `prime_header_org_signet_write()`, `PRIME_ORG_SIGNET`, `SECP256K1_KEY_PUB_LEN`, `secp256k1_public_get()`, `st_alloc()`, `st_avail_get()`, `st_cleanup`, `st_opt_get()`, `st_valid_destination()`, `st_wipe()`, and `st_write`.

Referenced by `prime_get()`.

5.379.1.6 size_t org_signet_length (prime_org_signet_t * org)

Definition at line 97 of file `orgs.c`.

Referenced by `org_signet_get()`.

5.379.1.7 prime_org_signet_t* org_signet_set (stringer_t * org)

Definition at line 150 of file `orgs.c`.

References `ed25519_public_set()`, `log_pedantic`, `org_signet_alloc()`, `org_signet_free()`, `org_signet_verify()`, `pl_data_get()`, `pl_length_get()`, `prime_field_get()`, `prime_field_t`, `prime_object_free()`, `prime_object_t`, `PRIME_ORG_SIGNET`, `prime_org_signet_t`, `prime_unpack()`, `secp256k1_public_set()`, `st_import()`, and `st_length_get()`.

Referenced by `prime_set()`.

5.379.1.8 bool_t org_signet_verify (prime_org_signet_t * org)

Definition at line 212 of file `orgs.c`.

References `ED25519_KEY_PUB_LEN`, `ed25519_public_get()`, `ed25519_verify()`, `MANAGEDBUF`, `prime_field_write()`, `PRIME_ORG_SIGNET`, `SECP256K1_KEY_PUB_LEN`, `secp256k1_public_get()`, `st_length_get()`, and `st_write`.

Referenced by `org_signet_set()`, and `prime_signet_validate()`.

5.380 src/providers/prime/keys/users.c File Reference

```
#include "magma.h"
```

Functions

- void [user_key_free](#) ([prime_user_key_t](#) *user)
- [prime_user_key_t](#) * [user_key_alloc](#) (void)
- [prime_user_key_t](#) * [user_key_generate](#) (void)
- size_t [user_key_length](#) ([prime_user_key_t](#) *user)
- [stringer_t](#) * [user_key_get](#) ([prime_user_key_t](#) *user, [stringer_t](#) *output)
- [prime_user_key_t](#) * [user_key_set](#) ([stringer_t](#) *user)
- [stringer_t](#) * [user_encrypted_key_get](#) ([stringer_t](#) *key, [prime_user_key_t](#) *user, [stringer_t](#) *output)

users.c

- [prime_user_key_t](#) * [user_encrypted_key_set](#) ([stringer_t](#) *key, [stringer_t](#) *user)

5.380.1 Function Documentation

5.380.1.1 [stringer_t](#)* [user_encrypted_key_get](#) ([stringer_t](#) *key, [prime_user_key_t](#) *user, [stringer_t](#) *output)

users.c

Definition at line 142 of file users.c.

References [aes_artifact_encrypt\(\)](#), [st_free\(\)](#), and [user_key_get\(\)](#).

Referenced by [prime_key_encrypt\(\)](#).

5.380.1.2 [prime_user_key_t](#)* [user_encrypted_key_set](#) ([stringer_t](#) *key, [stringer_t](#) *user)

Definition at line 159 of file users.c.

References [aes_artifact_decrypt\(\)](#), [prime_user_key_t](#), [st_free\(\)](#), and [user_key_set\(\)](#).

Referenced by [prime_key_decrypt\(\)](#).

5.380.1.3 [prime_user_key_t](#)* [user_key_alloc](#) (void)

Definition at line 21 of file users.c.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_user_key_t](#).

Referenced by [user_key_set\(\)](#).

5.380.1.4 void [user_key_free](#) ([prime_user_key_t](#) *user)

Definition at line 10 of file users.c.

References [ed25519_free\(\)](#), [mm_free\(\)](#), and [secp256k1_free\(\)](#).

Referenced by [prime_free\(\)](#), [user_key_generate\(\)](#), and [user_key_set\(\)](#).

5.380.1.5 `prime_user_key_t* user_key_generate (void)`

Definition at line 35 of file users.c.

References `ed25519_generate()`, `log_pedantic`, `mm_alloc()`, `prime_user_key_t`, `secp256k1_generate()`, and `user_key_free()`.

Referenced by `prime_key_generate()`.

5.380.1.6 `stringer_t* user_key_get (prime_user_key_t * user, stringer_t * output)`

Definition at line 61 of file users.c.

References `ED25519_KEY_PRIV_LEN`, `ed25519_private_get()`, `length`, `log_pedantic`, `MANAGEDBUF`, `prime_field_write()`, `prime_header_user_key_write()`, `PRIME_USER_KEY`, `SECP256K1_KEY_PRIV_LEN`, `secp256k1_private_get()`, `st_alloc()`, `st_avail_get()`, `st_cleanup`, `st_opt_get()`, `st_valid_destination()`, `st_wipe()`, `st_write`, and `user_key_length()`.

Referenced by `prime_get()`, and `user_encrypted_key_get()`.

5.380.1.7 `size_t user_key_length (prime_user_key_t * user)`

Definition at line 52 of file users.c.

Referenced by `user_key_get()`.

5.380.1.8 `prime_user_key_t* user_key_set (stringer_t * user)`

Definition at line 100 of file users.c.

References `ed25519_private_set()`, `log_pedantic`, `prime_field_get()`, `prime_field_t`, `prime_object_free()`, `prime_object_t`, `prime_unpack()`, `PRIME_USER_KEY`, `prime_user_key_t`, `secp256k1_private_set()`, `user_key_alloc()`, and `user_key_free()`.

Referenced by `prime_set()`, and `user_encrypted_key_set()`.

5.381 src/providers/prime/signets/users.c File Reference

```
#include "magma.h"
```

Functions

- void [user_signet_free](#) ([prime_user_signet_t](#) *user)
- [prime_user_signet_t](#) * [user_signet_alloc](#) (void)

users.c

- size_t [user_signet_length](#) ([prime_user_signet_t](#) *user)
- [stringer_t](#) * [user_signet_get](#) ([prime_user_signet_t](#) *user, [stringer_t](#) *output)
- [prime_user_signet_t](#) * [user_signet_set](#) ([stringer_t](#) *user)
- [stringer_t](#) * [user_signet_fingerprint](#) ([prime_user_signet_t](#) *user, [stringer_t](#) *output)
- bool_t [user_signet_verify_chain_of_custody](#) ([prime_user_signet_t](#) *user, [prime_user_signet_t](#) *previous)
- bool_t [user_signet_verify_org](#) ([prime_user_signet_t](#) *user, [prime_org_signet_t](#) *org)
- bool_t [user_signet_verify_self](#) ([prime_user_signet_t](#) *user)

5.381.1 Function Documentation

5.381.1.1 [prime_user_signet_t](#)* [user_signet_alloc](#) (void)

users.c

Definition at line 26 of file *users.c*.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_user_signet_t](#).

Referenced by [user_request_set\(\)](#), and [user_signet_set\(\)](#).

5.381.1.2 [stringer_t](#)* [user_signet_fingerprint](#) ([prime_user_signet_t](#) * *user*, [stringer_t](#) * *output*)

Definition at line 197 of file *users.c*.

References [ED25519_KEY_PUB_LEN](#), [ed25519_public_get\(\)](#), [ED25519_SIGNATURE_LEN](#), [hash_sha512\(\)](#), [MANAGEDBUF](#), [prime_field_write\(\)](#), [PRIME_USER_SIGNET](#), [SECP256K1_KEY_PUB_LEN](#), [secp256k1_public_get\(\)](#), [st_length_get\(\)](#), and [st_write](#).

Referenced by [prime_signet_fingerprint\(\)](#).

5.381.1.3 void [user_signet_free](#) ([prime_user_signet_t](#) * *user*)

Definition at line 12 of file *users.c*.

References [ed25519_free\(\)](#), [mm_free\(\)](#), [secp256k1_free\(\)](#), and [st_free\(\)](#).

Referenced by [prime_free\(\)](#), [user_request_generate\(\)](#), [user_request_rotation\(\)](#), [user_request_set\(\)](#), [user_request_sign\(\)](#), and [user_signet_set\(\)](#).

5.381.1.4 [stringer_t](#)* [user_signet_get](#) ([prime_user_signet_t](#) * *user*, [stringer_t](#) * *output*)

Definition at line 53 of file *users.c*.

References [ED25519_KEY_PUB_LEN](#), [ed25519_public_get\(\)](#), [ED25519_SIGNATURE_LEN](#), [length](#), [log_pedantic](#), [MANAGEDBUF](#), [prime_field_write\(\)](#), [prime_header_user_signet_write\(\)](#), [PRIME_USER_SIGNET](#), [SECP256K1_KEY_PUB_LEN](#), [secp256k1_public_get\(\)](#), [st_alloc\(\)](#), [st_avail_get\(\)](#), [st_cleanup](#), [st_opt_get\(\)](#), [st_valid_destination\(\)](#), [st_wipe\(\)](#), [st_write](#), and [user_signet_length\(\)](#).

Referenced by [prime_get\(\)](#).

5.381.1.5 size_t user_signet_length (prime_user_signet_t * user)

Definition at line 40 of file users.c.

Referenced by user_signet_get().

5.381.1.6 prime_user_signet_t* user_signet_set (stringer_t * user)

Definition at line 114 of file users.c.

References ed25519_public_set(), log_pedantic, pl_data_get(), pl_length_get(), prime_field_get(), prime_field_t, prime_object_free(), prime_object_t, prime_unpack(), PRIME_USER_SIGNET, prime_user_signet_t, secp256k1_public_set(), st_import(), st_length_get(), user_signet_alloc(), user_signet_free(), and user_signet_verify_self().

Referenced by prime_set().

5.381.1.7 bool_t user_signet_verify_chain_of_custody (prime_user_signet_t * user, prime_user_signet_t * previous)

Definition at line 221 of file users.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate().

5.381.1.8 bool_t user_signet_verify_org (prime_user_signet_t * user, prime_org_signet_t * org)

Definition at line 249 of file users.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate().

5.381.1.9 bool_t user_signet_verify_self (prime_user_signet_t * user)

Definition at line 279 of file users.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate(), and user_signet_set().

5.382 src/providers/prime/messages/chunks/chunks.c File Reference

```
#include "magma.h"
```

Functions

- [int32_t chunk_buffer_size](#) ([stringer_t](#) *chunk)
- [int32_t chunk_header_size](#) ([stringer_t](#) *chunk)
- [prime_message_chunk_type_t chunk_header_type](#) ([stringer_t](#) *chunk)
- [int_t chunk_header_read](#) ([stringer_t](#) *data, [uint8_t](#) *type, [uint32_t](#) *size, [placer_t](#) *chunk)
- [int_t chunk_buffer_read](#) ([stringer_t](#) *data, [uint8_t](#) *type, [uint32_t](#) *payload_size, [uint32_t](#) *buffer_size, [placer_t](#) *chunk)

[chunks.c](#)

- [stringer_t](#) * [chunk_header_write](#) ([prime_message_chunk_type_t](#) type, [size_t](#) size, [stringer_t](#) *output)

5.382.1 Function Documentation

5.382.1.1 [int_t chunk_buffer_read](#) ([stringer_t](#) * *data*, [uint8_t](#) * *type*, [uint32_t](#) * *payload_size*, [uint32_t](#) * *buffer_size*, [placer_t](#) * *chunk*)

[chunks.c](#)

Definition at line 163 of file [chunks.c](#).

References [chunk_header_read\(\)](#), [log_pedantic](#), [pl_length_get\(\)](#), [PRIME_CHUNK_EPHEMERAL](#), [PRIME_CHUNK_INVALID](#), [SECP256K1_SHARED_SECRET_LEN](#), [slots_count\(\)](#), and [st_length_get\(\)](#).

Referenced by [part_decrypt\(\)](#).

5.382.1.2 [int32_t chunk_buffer_size](#) ([stringer_t](#) * *chunk*)

Definition at line 10 of file [chunks.c](#).

References [chunk_header_type\(\)](#), *data*, [log_pedantic](#), [mm_copy\(\)](#), [PRIME_CHUNK_EPHEMERAL](#), [PRIME_CHUNK_INVALID](#), [PRIME_SIGNATURE_TREE](#), [SECP256K1_SHARED_SECRET_LEN](#), [slots_count\(\)](#), [st_empty_out\(\)](#), and *type*.

5.382.1.3 [int_t chunk_header_read](#) ([stringer_t](#) * *data*, [uint8_t](#) * *type*, [uint32_t](#) * *size*, [placer_t](#) * *chunk*)

Definition at line 127 of file [chunks.c](#).

References [chunk_header_size\(\)](#), [chunk_header_type\(\)](#), [log_pedantic](#), [pl_init\(\)](#), [pl_length_get\(\)](#), [PRIME_CHUNK_EPHEMERAL](#), [PRIME_CHUNK_INVALID](#), [PRIME_SIGNATURE_TREE](#), [SECP256K1_SHARED_SECRET_LEN](#), [slots_count\(\)](#), [st_data_get\(\)](#), and [st_length_get\(\)](#).

Referenced by [chunk_buffer_read\(\)](#), and [naked_message_get\(\)](#).

5.382.1.4 [int32_t chunk_header_size](#) ([stringer_t](#) * *chunk*)

Definition at line 40 of file [chunks.c](#).

References [chunk_header_type\(\)](#), *data*, [log_pedantic](#), [mm_copy\(\)](#), [PRIME_CHUNK_INVALID](#), [PRIME_SIGNATURE_TREE](#), [st_empty_out\(\)](#), and *type*.

Referenced by [chunk_header_read\(\)](#), [encrypted_chunk_get\(\)](#), and [ephemeral_chunk_set\(\)](#).

5.382.1.5 `prime_message_chunk_type_t chunk_header_type (stringer_t * chunk)`

Definition at line 65 of file chunks.c.

References `data`, `log_pedantic`, `PRIME_CHUNK_BODY`, `PRIME_CHUNK_COMMON`, `PRIME_CHUNK_DESTINATION`, `PRIME_CHUNK_EPHEMERAL`, `PRIME_CHUNK_HEADERS`, `PRIME_CHUNK_INVALID`, `PRIME_CHUNK_ORIGIN`, `PRIME_SIGNATURE_DESTINATION`, `PRIME_SIGNATURE_ORGIN`, `PRIME_SIGNATURE_TREE`, `PRIME_SIGNATURE_USER`, `st_empty_out()`, and `type()`.

Referenced by `chunk_buffer_size()`, `chunk_header_read()`, `chunk_header_size()`, `encrypted_chunk_get()`, `ephemeral_chunk_set()`, `signature_full_verify()`, and `signature_tree_verify()`.

5.382.1.6 `stringer_t* chunk_header_write (prime_message_chunk_type_t type, size_t size, stringer_t * output)`

Definition at line 189 of file chunks.c.

References `log_error`, `log_pedantic`, `mm_copy()`, `PRIME_MAX_3_BYTE`, `st_data_get()`, `st_length_set()`, `st_opt_get()`, `st_output()`, and `st_valid_tracked()`.

Referenced by `ephemeral_chunk_get()`.

5.383 src/providers/prime/messages/chunks/chunks.h File Reference

Functions

- [int_t chunk_buffer_read](#) ([stringer_t](#) *data, [uint8_t](#) *type, [uint32_t](#) *payload_size, [uint32_t](#) *buffer_size, [placer_t](#) *chunk)

chunks.c

- [int32_t chunk_buffer_size](#) ([stringer_t](#) *chunk)
- [int_t chunk_header_read](#) ([stringer_t](#) *data, [uint8_t](#) *type, [uint32_t](#) *size, [placer_t](#) *chunk)
- [int32_t chunk_header_size](#) ([stringer_t](#) *chunk)
- [prime_message_chunk_type_t chunk_header_type](#) ([stringer_t](#) *chunk)
- [stringer_t](#) * [chunk_header_write](#) ([prime_message_chunk_type_t](#) type, [size_t](#) size, [stringer_t](#) *output)
- [prime_chunk_keks_t](#) * [keks_alloc](#) (void)

keks.c

- [void keks_cleanup](#) ([prime_chunk_keks_t](#) *keks)
 - [void keks_free](#) ([prime_chunk_keks_t](#) *keks)
 - [prime_chunk_keks_t](#) * [keks_get](#) ([prime_chunk_keys_t](#) *keys, [prime_chunk_keks_t](#) *keks)
- Uses the private ephemeral encryption key to compute a KEK with all of the provided actor public keys.*

- [prime_chunk_keks_t](#) * [keks_set](#) ([prime_chunk_keys_t](#) *keys, [prime_chunk_keks_t](#) *keks)
- Uses the public ephemeral encryption key to compute a KEK with all of the provided actor private keys.*

- [int_t slots_actors](#) ([prime_message_chunk_type_t](#) type)

slots.c

- [prime_chunk_slots_t](#) * [slots_alloc](#) ([prime_message_chunk_type_t](#) type)
- [placer_t](#) [slots_buffer](#) ([prime_chunk_slots_t](#) *slots)
- [void slots_cleanup](#) ([prime_chunk_slots_t](#) *slots)
- [int_t slots_count](#) ([prime_message_chunk_type_t](#) type)

Returns the number of slots for a given chunk type.

- [void slots_free](#) ([prime_chunk_slots_t](#) *slots)
- [stringer_t](#) * [slots_get](#) ([prime_message_chunk_type_t](#) type, [stringer_t](#) *slots, [prime_chunk_keks_t](#) *keks, [stringer_t](#) *output)
- [stringer_t](#) * [slots_key](#) ([prime_chunk_slots_t](#) *slots, [prime_chunk_keks_t](#) *keks, [stringer_t](#) *output)

Get the unstretched key value using the first available kek with a matching keyslot value.

- [prime_chunk_slots_t](#) * [slots_set](#) ([prime_message_chunk_type_t](#) type, [stringer_t](#) *key, [prime_chunk_keks_t](#) *keks)
- [stringer_t](#) * [signature_full_get](#) ([prime_message_chunk_type_t](#) type, [ed25519_key_t](#) *signing, [prime_chunk_keks_t](#) *keks, [stringer_t](#) *data)

signature.c

- [int_t signature_full_verify](#) ([ed25519_key_t](#) *signing, [prime_chunk_keks_t](#) *keks, [stringer_t](#) *data, [stringer_t](#) *chunk)

Verify an Ed25519 signature using the EdDSA algorithm.

- [int_t signature_tree_add](#) ([prime_signature_tree_t](#) *chunk, [stringer_t](#) *data)
- [prime_signature_tree_t](#) * [signature_tree_alloc](#) (void)
- [void signature_tree_cleanup](#) ([prime_signature_tree_t](#) *chunk)
- [void signature_tree_free](#) ([prime_signature_tree_t](#) *chunk)
- [stringer_t](#) * [signature_tree_get](#) ([ed25519_key_t](#) *signing, [prime_signature_tree_t](#) *chunk, [prime_chunk_keks_t](#) *keks)

Calculates the tree signature value by concatenating the hashes together, and generating an Ed25519 signature using the result.

- [int_t signature_tree_verify](#) ([ed25519_key_t](#) *signing, [prime_signature_tree_t](#) *chunk, [prime_chunk_keks_t](#) *keks, [stringer_t](#) *data)

Verify an Ed25519 signature using the EdDSA algorithm.

- `prime_ephemeral_chunk_t * ephemeral_chunk_alloc` (void)

ephemeral.c

- `stringer_t * ephemeral_chunk_buffer` (`prime_ephemeral_chunk_t *chunk`)
- `void ephemeral_chunk_cleanup` (`prime_ephemeral_chunk_t *chunk`)
- `void ephemeral_chunk_free` (`prime_ephemeral_chunk_t *chunk`)
- `prime_ephemeral_chunk_t * ephemeral_chunk_get` (`ed25519_key_t *signing`, `secp256k1_key_t *encryption`)

Generate an ephemeral message chunk. A public encryption key is required, while the public signing key is optional, but will be included if a signing key is provided.

- `prime_ephemeral_chunk_t * ephemeral_chunk_set` (`stringer_t *chunk`)

Parse an ephemeral message chunk. A public encryption key is required, while the ephemeral signing key is optional.

- `prime_encrypted_chunk_t * encrypted_chunk_alloc` (void)

encrypted.c

- `stringer_t * encrypted_chunk_buffer` (`prime_encrypted_chunk_t *chunk`)

Returns the chunk buffer, which contains a serialized version of the encrypted chunk data.

- `void encrypted_chunk_cleanup` (`prime_encrypted_chunk_t *chunk`)
- `void encrypted_chunk_free` (`prime_encrypted_chunk_t *chunk`)
- `stringer_t * encrypted_chunk_get` (`ed25519_key_t *signing`, `prime_chunk_keks_t *keks`, `stringer_t *chunk`, `stringer_t *output`, `bool_t *spanning`)

Take an encrypted message chunk, in serialized form, and return the decrypted payload data.

- `prime_encrypted_chunk_t * encrypted_chunk_set` (`prime_message_chunk_type_t type`, `ed25519_key_t *signing`, `prime_chunk_keks_t *keks`, `prime_message_chunk_flags_t flags`, `stringer_t *payload`)

Generate an encrypted message chunk. The signing and encryption keys are required, along with the public encryption key for at least one actor.

5.383.1 Function Documentation

5.383.1.1 `int_t chunk_buffer_read` (`stringer_t * data`, `uint8_t * type`, `uint32_t * payload_size`, `uint32_t * buffer_size`, `placer_t * chunk`)

chunks.c

Definition at line 163 of file *chunks.c*.

References `chunk_header_read()`, `log_pedantic`, `pl_length_get()`, `PRIME_CHUNK_EPHEMERAL`, `PRIME_CHUNK_INVALID`, `SECP256K1_SHARED_SECRET_LEN`, `slots_count()`, and `st_length_get()`.

Referenced by `part_decrypt()`.

5.383.1.2 `int32_t chunk_buffer_size` (`stringer_t * chunk`)

Definition at line 10 of file *chunks.c*.

References `chunk_header_type()`, `data`, `log_pedantic`, `mm_copy()`, `PRIME_CHUNK_EPHEMERAL`, `PRIME_CHUNK_INVALID`, `PRIME_SIGNATURE_TREE`, `SECP256K1_SHARED_SECRET_LEN`, `slots_count()`, `st_empty_out()`, and `type()`.

5.383.1.3 int_t chunk_header_read (stringer_t * data, uint8_t * type, uint32_t * size, placer_t * chunk)

Definition at line 127 of file chunks.c.

References chunk_header_size(), chunk_header_type(), log_pedantic, pl_init(), pl_length_get(), PRIME_CHUNK_EPHEMERAL, PRIME_CHUNK_INVALID, PRIME_SIGNATURE_TREE, SECP256K1_SHARED_SECRET_LEN, slots_count(), st_data_get(), and st_length_get().

Referenced by chunk_buffer_read(), and naked_message_get().

5.383.1.4 int32_t chunk_header_size (stringer_t * chunk)

Definition at line 40 of file chunks.c.

References chunk_header_type(), data, log_pedantic, mm_copy(), PRIME_CHUNK_INVALID, PRIME_SIGNATURE_TREE, st_empty_out(), and type().

Referenced by chunk_header_read(), encrypted_chunk_get(), and ephemeral_chunk_set().

5.383.1.5 prime_message_chunk_type_t chunk_header_type (stringer_t * chunk)

Definition at line 65 of file chunks.c.

References data, log_pedantic, PRIME_CHUNK_BODY, PRIME_CHUNK_COMMON, PRIME_CHUNK_DESTINATION, PRIME_CHUNK_EPHEMERAL, PRIME_CHUNK_HEADERS, PRIME_CHUNK_INVALID, PRIME_CHUNK_ORIGIN, PRIME_SIGNATURE_DESTINATION, PRIME_SIGNATURE_ORGIN, PRIME_SIGNATURE_TREE, PRIME_SIGNATURE_USER, st_empty_out(), and type().

Referenced by chunk_buffer_size(), chunk_header_read(), chunk_header_size(), encrypted_chunk_get(), ephemeral_chunk_set(), signature_full_verify(), and signature_tree_verify().

5.383.1.6 stringer_t* chunk_header_write (prime_message_chunk_type_t type, size_t size, stringer_t * output)

Definition at line 189 of file chunks.c.

References log_error, log_pedantic, mm_copy(), PRIME_MAX_3_BYTE, st_data_get(), st_length_set(), st_opt_get(), st_output(), and st_valid_tracked().

Referenced by ephemeral_chunk_get().

5.383.1.7 prime_encrypted_chunk_t* encrypted_chunk_alloc (void)

[encrypted.c](#)

Definition at line 44 of file encrypted.c.

References log_pedantic, mm_alloc(), mm_wipe(), and prime_encrypted_chunk_t.

Referenced by encrypted_chunk_set().

5.383.1.8 stringer_t* encrypted_chunk_buffer (prime_encrypted_chunk_t * chunk)

Returns the chunk buffer, which contains a serialized version of the encrypted chunk data.

Definition at line 62 of file encrypted.c.

Referenced by naked_message_set(), and part_buffer().

5.383.1.9 void encrypted_chunk_cleanup (prime_encrypted_chunk_t * chunk)

Definition at line 37 of file encrypted.c.

References encrypted_chunk_free().

Referenced by part_encrypt().

5.383.1.10 void encrypted_chunk_free (prime_encrypted_chunk_t * chunk)

Definition at line 9 of file encrypted.c.

References log_pedantic, mm_free(), prime_encrypted_chunk_t, slots_free(), and st_free().

Referenced by encrypted_chunk_cleanup(), encrypted_chunk_set(), and encrypted_message_free().

5.383.1.11 stringer_t* encrypted_chunk_get (ed25519_key_t * signing, prime_chunk_keks_t * keks, stringer_t * chunk, stringer_t * output, bool_t * spanning)

Take an encrypted message chunk, in serialized form, and return the decrypted payload data.

Definition at line 175 of file encrypted.c.

References aes_chunk_decrypt(), chunk_header_size(), chunk_header_type(), ED25519_PUB, ED25519_SIGNATURE_LEN, ed25519_type(), ed25519_verify(), hash_sha512(), log_pedantic, MANAGEDBUF, mm_copy(), PLACER, PRIME_CHUNK_FLAG_SPANNING, PRIME_CHUNK_INVALID, SECP256K1_SHARED_SECRET_LEN, slots_count(), slots_get(), st_cmp_cs_eq(), st_data_get(), st_free(), st_length_get(), st_output(), st_set(), st_write, and type().

Referenced by naked_message_get(), and part_decrypt().

5.383.1.12 prime_encrypted_chunk_t* encrypted_chunk_set (prime_message_chunk_type_t type, ed25519_key_t * signing, prime_chunk_keks_t * keks, prime_message_chunk_flags_t flags, stringer_t * payload)

Generate an encrypted message chunk. The signing and encryption keys are required, along with the public encryption key for at least one actor.

Definition at line 77 of file encrypted.c.

References aes_chunk_encrypt(), ED25519_PRIV, ed25519_sign(), ed25519_type(), encrypted_chunk_alloc(), encrypted_chunk_free(), hash_sha512(), log_pedantic, MANAGED, MANAGEDBUF, mm_copy(), PLACER, prime_chunk_slots_t, prime_encrypted_chunk_t, rand_write(), slots_free(), slots_set(), st_alloc(), st_append_out(), st_data_get(), st_length_get(), st_length_set(), and st_set().

Referenced by naked_message_set(), and part_encrypt().

5.383.1.13 prime_ephemeral_chunk_t* ephemeral_chunk_alloc (void)

[ephemeral.c](#)

Definition at line 34 of file ephemeral.c.

References log_pedantic, mm_alloc(), mm_wipe(), and prime_ephemeral_chunk_t.

Referenced by ephemeral_chunk_get(), and ephemeral_chunk_set().

5.383.1.14 stringer_t* ephemeral_chunk_buffer (prime_ephemeral_chunk_t * chunk)

Definition at line 49 of file ephemeral.c.

Referenced by naked_message_set().

5.383.1.15 void ephemeral_chunk_cleanup (prime_ephemeral_chunk_t * *chunk*)

Definition at line 27 of file ephemeral.c.

References ephemeral_chunk_free().

Referenced by naked_message_get().

5.383.1.16 void ephemeral_chunk_free (prime_ephemeral_chunk_t * *chunk*)

Definition at line 10 of file ephemeral.c.

References ed25519_free(), log_pedantic, mm_free(), secp256k1_free(), and st_free().

Referenced by encrypted_message_free(), ephemeral_chunk_cleanup(), ephemeral_chunk_get(), ephemeral_chunk_set(), and naked_message_get().

5.383.1.17 prime_ephemeral_chunk_t* ephemeral_chunk_get (ed25519_key_t * *signing*, secp256k1_key_t * *encryption*)

Generate an ephemeral message chunk. A public encryption key is required, while the public signing key is optional, but will be included if a signing key is provided.

Definition at line 64 of file ephemeral.c.

References chunk_header_write(), ED25519_KEY_PUB_LEN, ED25519_PRIV, ED25519_PUB, ed25519_public_get(), ed25519_public_set(), ephemeral_chunk_alloc(), ephemeral_chunk_free(), log_pedantic, MANAGEDBUF, PRIME_CHUNK_EPHEMERAL, prime_ephemeral_chunk_t, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), secp256k1_public_set(), st_length_get(), st_merge, USER_ENCRYPTION_KEY, and USER_SIGNING_KEY.

Referenced by naked_message_set().

5.383.1.18 prime_ephemeral_chunk_t* ephemeral_chunk_set (stringer_t * *chunk*)

Parse an ephemeral message chunk. A public encryption key is required, while the ephemeral signing key is optional.

Definition at line 111 of file ephemeral.c.

References chunk_header_size(), chunk_header_type(), data, ed25519_public_set(), ephemeral_chunk_alloc(), ephemeral_chunk_free(), FOREIGNDATA, HEAP, JOINTED, log_pedantic, pl_data_get(), pl_init(), PLACER, PLACER_T, placer_t, PRIME_CHUNK_EPHEMERAL, prime_ephemeral_chunk_t, secp256k1_public_set(), st_alloc_opts(), st_data_get(), st_data_set(), st_empty_out(), st_length_get(), and st_length_set().

Referenced by naked_message_get().

5.383.1.19 prime_chunk_keks_t* keks_alloc (void)

[keks.c](#)

Definition at line 34 of file keks.c.

References log_pedantic, mm_alloc(), mm_wipe(), and prime_chunk_keks_t.

Referenced by keks_get(), and keks_set().

5.383.1.20 void keks_cleanup (prime_chunk_keks_t * *keks*)

Definition at line 25 of file keks.c.

References keks_free().

5.383.1.21 void keks_free (prime_chunk_keks_t * keks)

Definition at line 10 of file keks.c.

References log_pedantic, mm_free(), and st_cleanup.

Referenced by keks_cleanup(), keks_get(), keks_set(), and naked_message_get().

5.383.1.22 prime_chunk_keks_t* keks_get (prime_chunk_keys_t * keys, prime_chunk_keks_t * keks)

Uses the private ephemeral encryption key to compute a KEK with all of the provided actor public keys.

Definition at line 53 of file keks.c.

References keks_alloc(), keks_free(), prime_chunk_keks_t, secp256k1_compute_kek(), SECP256K1_PRIV, SECP256K1_PUB, and secp256k1_type().

Referenced by naked_message_set().

5.383.1.23 prime_chunk_keks_t* keks_set (prime_chunk_keys_t * keys, prime_chunk_keks_t * keks)

Uses the public ephemeral encryption key to compute a KEK with all of the provided actor private keys.

Definition at line 98 of file keks.c.

References keks_alloc(), keks_free(), prime_chunk_keks_t, secp256k1_compute_kek(), SECP256K1_PRIV, SECP256K1_PUB, and secp256k1_type().

Referenced by naked_message_get().

5.383.1.24 stringer_t* signature_full_get (prime_message_chunk_type_t type, ed25519_key_t * signing, prime_chunk_keks_t * keks, stringer_t * data)

[signature.c](#)

Definition at line 191 of file signature.c.

References ED25519_PRIV, ed25519_sign(), ED25519_SIGNATURE_LEN, ed25519_type(), hash_sha512(), MANAGEDBUF, PLACER, placer_t, prime_chunk_slots_t, rand_write(), slots_buffer(), slots_free(), slots_set(), st_alloc(), st_cleanup, st_length_get(), st_write, and st_xor().

Referenced by naked_message_set().

5.383.1.25 int_t signature_full_verify (ed25519_key_t * signing, prime_chunk_keks_t * keks, stringer_t * data, stringer_t * chunk)

Verify an Ed25519 signature using the EdDSA algorithm.

Returns:

0 for successful signature verification, -1 for a signature verification failures, -2 for processing or parameter issue.

Definition at line 234 of file signature.c.

References chunk_header_type(), ED25519_PRIV, ED25519_PUB, ED25519_SIGNATURE_LEN, ed25519_type(), ed25519_verify(), hash_sha512(), MANAGEDBUF, pl_init(), placer_t, PRIME_SIGNATURE_USER, SECP256K1_SHARED_SECRET_LEN, slots_count(), slots_get(), st_data_get(), st_length_get(), st_xor(), and type().

Referenced by naked_message_get().

5.383.1.26 int_t signature_tree_add (prime_signature_tree_t * *chunk*, stringer_t * *data*)

Definition at line 52 of file signature.c.

References hash_sha512(), inx_count(), inx_insert(), M_TYPE_UINT64, st_free(), st_length_get(), multi_t::type, multi_t::u64, and multi_t::val.

Referenced by naked_message_get(), and naked_message_set().

5.383.1.27 prime_signature_tree_t* signature_tree_alloc (void)

Definition at line 32 of file signature.c.

References inx_alloc(), log_pedantic, M_INX_LINKED, mm_alloc(), mm_wipe(), prime_signature_tree_t, and st_free().

Referenced by naked_message_get(), and naked_message_set().

5.383.1.28 void signature_tree_cleanup (prime_signature_tree_t * *chunk*)

Definition at line 25 of file signature.c.

References signature_tree_free().

5.383.1.29 void signature_tree_free (prime_signature_tree_t * *chunk*)

Definition at line 10 of file signature.c.

References inx_free(), log_pedantic, and mm_free().

Referenced by naked_message_get(), naked_message_set(), and signature_tree_cleanup().

5.383.1.30 stringer_t* signature_tree_get (ed25519_key_t * *signing*, prime_signature_tree_t * *chunk*, prime_chunk_keks_t * *keks*)

Calculates the tree signature value by concatenating the hashes together, and generating an Ed25519 signature using the result.

Definition at line 79 of file signature.c.

References count, ED25519_PRIV, ed25519_sign(), ed25519_type(), hash_sha512(), HEAP, inx_count(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), JOINTED, MANAGED_T, MANAGEDBUF, PLACER, placer_t, prime_chunk_slots_t, PRIME_SIGNATURE_TREE, rand_write(), slots_buffer(), slots_free(), slots_set(), st_alloc(), st_alloc_opts(), st_append, st_cleanup, st_free(), st_length_get(), st_write, st_xor(), and type().

Referenced by naked_message_set().

5.383.1.31 int_t signature_tree_verify (ed25519_key_t * *signing*, prime_signature_tree_t * *chunk*, prime_chunk_keks_t * *keks*, stringer_t * *data*)

Verify an Ed25519 signature using the EdDSA algorithm.

Returns:

0 for successful signature verification, -1 for a signature verification failures, -2 for processing or parameter issue.

Definition at line 146 of file signature.c.

References chunk_header_type(), count, ED25519_PRIV, ED25519_PUB, ED25519_SIGNATURE_LEN, ed25519_type(), ed25519_verify(), hash_sha512(), HEAP, inx_count(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), JOINTED, MANAGED_T, MANAGEDBUF, pl_init(), placer_t, PRIME_SIGNATURE_TREE, SECP256K1_SHARED_SECRET_LEN, slots_count(), slots_get(), st_alloc_opts(), st_append, st_data_get(), st_free(), st_length_get(), and st_xor().

Referenced by naked_message_get().

5.383.1.32 int_t slots_actors (prime_message_chunk_type_t type)[slots.c](#)

Definition at line 99 of file slots.c.

References `log_pedantic`, `PRIME_ACTOR_AUTHOR`, `PRIME_ACTOR_DESTINATION`, `PRIME_ACTOR_NONE`, `PRIME_ACTOR_ORIGIN`, `PRIME_ACTOR_RECIPIENT`, `PRIME_CHUNK_BODY`, `PRIME_CHUNK_COMMON`, `PRIME_CHUNK_DESTINATION`, `PRIME_CHUNK_HEADERS`, `PRIME_CHUNK_ORIGIN`, `PRIME_SIGNATURE_DESTINATION`, `PRIME_SIGNATURE_ORGIN`, `PRIME_SIGNATURE_TREE`, and `PRIME_SIGNATURE_USER`.

Referenced by `slots_alloc()`, `slots_count()`, `slots_get()`, and `slots_set()`.

5.383.1.33 prime_chunk_slots_t* slots_alloc (prime_message_chunk_type_t type)

Definition at line 39 of file slots.c.

References `count`, `log_pedantic`, `mm_alloc()`, `mm_wipe()`, `pl_init()`, `PRIME_ACTOR_AUTHOR`, `PRIME_ACTOR_DESTINATION`, `PRIME_ACTOR_NONE`, `PRIME_ACTOR_ORIGIN`, `PRIME_ACTOR_RECIPIENT`, `prime_chunk_slots_t`, `SECP256K1_SHARED_SECRET_LEN`, `slots_actors()`, `slots_count()`, and `slots_free()`.

Referenced by `slots_get()`, and `slots_set()`.

5.383.1.34 placer_t slots_buffer (prime_chunk_slots_t * slots)

Definition at line 156 of file slots.c.

References `pl_length_get()`, `pl_null()`, and `placer_t`.

Referenced by `signature_full_get()`, and `signature_tree_get()`.

5.383.1.35 void slots_cleanup (prime_chunk_slots_t * slots)

Definition at line 30 of file slots.c.

References `slots_free()`.

5.383.1.36 int_t slots_count (prime_message_chunk_type_t type)

Returns the number of slots for a given chunk type.

Definition at line 152 of file slots.c.

References `bitwise_count()`, and `slots_actors()`.

Referenced by `chunk_buffer_read()`, `chunk_buffer_size()`, `chunk_header_read()`, `encrypted_chunk_get()`, `signature_full_verify()`, `signature_tree_verify()`, `slots_alloc()`, `slots_get()`, and `slots_set()`.

5.383.1.37 void slots_free (prime_chunk_slots_t * slots)

Definition at line 16 of file slots.c.

References `log_pedantic`, and `mm_free()`.

Referenced by `encrypted_chunk_free()`, `encrypted_chunk_set()`, `signature_full_get()`, `signature_tree_get()`, `slots_alloc()`, `slots_cleanup()`, `slots_get()`, and `slots_set()`.

5.383.1.38 stringer_t* slots_get (prime_message_chunk_type_t *type*, stringer_t * *slots*, prime_chunk_keks_t * *keks*, stringer_t * *output*)

Definition at line 273 of file slots.c.

References count, log_pedantic, pl_init(), PRIME_ACTOR_AUTHOR, PRIME_ACTOR_DESTINATION, PRIME_ACTOR_NONE, PRIME_ACTOR_ORIGIN, PRIME_ACTOR_RECIPIENT, prime_chunk_slots_t, SECP256K1_SHARED_SECRET_LEN, slots_actors(), slots_alloc(), slots_count(), slots_free(), slots_key(), st_data_get(), and st_length_get().

Referenced by encrypted_chunk_get(), signature_full_verify(), and signature_tree_verify().

5.383.1.39 stringer_t* slots_key (prime_chunk_slots_t * *slots*, prime_chunk_keks_t * *keks*, stringer_t * *output*)

Get the unstretched key value using the first available kek with a matching keyslot value.

Definition at line 170 of file slots.c.

References log_pedantic, MANAGEDBUF, pl_empty(), SECP256K1_SHARED_SECRET_LEN, st_cmp_cs_eq(), st_free(), st_length_get(), st_output(), st_set(), and st_xor().

Referenced by slots_get().

5.383.1.40 prime_chunk_slots_t* slots_set (prime_message_chunk_type_t *type*, stringer_t * *key*, prime_chunk_keks_t * *keks*)

Definition at line 237 of file slots.c.

References PRIME_ACTOR_AUTHOR, PRIME_ACTOR_DESTINATION, PRIME_ACTOR_NONE, PRIME_ACTOR_ORIGIN, PRIME_ACTOR_RECIPIENT, prime_chunk_slots_t, slots_actors(), slots_alloc(), slots_count(), slots_free(), st_length_get(), and st_xor().

Referenced by encrypted_chunk_set(), signature_full_get(), and signature_tree_get().

5.384 src/providers/prime/messages/chunks/encrypted.c File Reference

```
#include "magma.h"
```

Functions

- void [encrypted_chunk_free](#) ([prime_encrypted_chunk_t](#) *chunk)
- void [encrypted_chunk_cleanup](#) ([prime_encrypted_chunk_t](#) *chunk)
- [prime_encrypted_chunk_t](#) * [encrypted_chunk_alloc](#) (void)
encrypted.c
- [stringer_t](#) * [encrypted_chunk_buffer](#) ([prime_encrypted_chunk_t](#) *chunk)
Returns the chunk buffer, which contains a serialized version of the encrypted chunk data.
- [prime_encrypted_chunk_t](#) * [encrypted_chunk_set](#) ([prime_message_chunk_type_t](#) type, [ed25519_key_t](#) *signing, [prime_chunk_keks_t](#) *keks, [prime_message_chunk_flags_t](#) flags, [stringer_t](#) *payload)
Generate an encrypted message chunk. The signing and encryption keys are required, along with the public encryption key for at least one actor.
- [stringer_t](#) * [encrypted_chunk_get](#) ([ed25519_key_t](#) *signing, [prime_chunk_keks_t](#) *keks, [stringer_t](#) *chunk, [stringer_t](#) *output, [bool_t](#) *spanning)
Take an encrypted message chunk, in serialized form, and return the decrypted payload data.

5.384.1 Function Documentation

5.384.1.1 [prime_encrypted_chunk_t](#)* [encrypted_chunk_alloc](#) (void)

[encrypted.c](#)

Definition at line 44 of file [encrypted.c](#).

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_encrypted_chunk_t](#).

Referenced by [encrypted_chunk_set\(\)](#).

5.384.1.2 [stringer_t](#)* [encrypted_chunk_buffer](#) ([prime_encrypted_chunk_t](#) * *chunk*)

Returns the chunk buffer, which contains a serialized version of the encrypted chunk data.

Definition at line 62 of file [encrypted.c](#).

Referenced by [naked_message_set\(\)](#), and [part_buffer\(\)](#).

5.384.1.3 void [encrypted_chunk_cleanup](#) ([prime_encrypted_chunk_t](#) * *chunk*)

Definition at line 37 of file [encrypted.c](#).

References [encrypted_chunk_free\(\)](#).

Referenced by [part_encrypt\(\)](#).

5.384.1.4 void [encrypted_chunk_free](#) ([prime_encrypted_chunk_t](#) * *chunk*)

Definition at line 9 of file [encrypted.c](#).

References [log_pedantic](#), [mm_free\(\)](#), [prime_encrypted_chunk_t](#), [slots_free\(\)](#), and [st_free\(\)](#).

Referenced by [encrypted_chunk_cleanup\(\)](#), [encrypted_chunk_set\(\)](#), and [encrypted_message_free\(\)](#).

5.384.1.5 stringer_t* encrypted_chunk_get (ed25519_key_t * *signing*, prime_chunk_keks_t * *keks*, stringer_t * *chunk*, stringer_t * *output*, bool_t * *spanning*)

Take an encrypted message chunk, in serialized form, and return the decrypted payload data.

Definition at line 175 of file encrypted.c.

References aes_chunk_decrypt(), chunk_header_size(), chunk_header_type(), ED25519_PUB, ED25519_SIGNATURE_LEN, ed25519_type(), ed25519_verify(), hash_sha512(), log_pedantic, MANAGEDBUF, mm_copy(), PLACER, PRIME_CHUNK_FLAG_SPANNING, PRIME_CHUNK_INVALID, SECP256K1_SHARED_SECRET_LEN, slots_count(), slots_get(), st_cmp_cs_eq(), st_data_get(), st_free(), st_length_get(), st_output(), st_set(), st_write, and type().

Referenced by naked_message_get(), and part_decrypt().

5.384.1.6 prime_encrypted_chunk_t* encrypted_chunk_set (prime_message_chunk_type_t *type*, ed25519_key_t * *signing*, prime_chunk_keks_t * *keks*, prime_message_chunk_flags_t *flags*, stringer_t * *payload*)

Generate an encrypted message chunk. The signing and encryption keys are required, along with the public encryption key for at least one actor.

Definition at line 77 of file encrypted.c.

References aes_chunk_encrypt(), ED25519_PRIV, ed25519_sign(), ed25519_type(), encrypted_chunk_alloc(), encrypted_chunk_free(), hash_sha512(), log_pedantic, MANAGED, MANAGEDBUF, mm_copy(), PLACER, prime_chunk_slots_t, prime_encrypted_chunk_t, rand_write(), slots_free(), slots_set(), st_alloc(), st_append_out(), st_data_get(), st_length_get(), st_length_set(), and st_set().

Referenced by naked_message_set(), and part_encrypt().

5.385 src/providers/prime/messages/chunks/ephemeral.c File Reference

```
#include "magma.h"
```

Functions

- void [ephemeral_chunk_free](#) ([prime_ephemeral_chunk_t](#) *chunk)
- void [ephemeral_chunk_cleanup](#) ([prime_ephemeral_chunk_t](#) *chunk)
- [prime_ephemeral_chunk_t](#) * [ephemeral_chunk_alloc](#) (void)

[ephemeral.c](#)

- [stringer_t](#) * [ephemeral_chunk_buffer](#) ([prime_ephemeral_chunk_t](#) *chunk)
- [prime_ephemeral_chunk_t](#) * [ephemeral_chunk_get](#) ([ed25519_key_t](#) *signing, [secp256k1_key_t](#) *encryption)

Generate an ephemeral message chunk. A public encryption key is required, while the public signing key is optional, but will be included if a signing key is provided.

- [prime_ephemeral_chunk_t](#) * [ephemeral_chunk_set](#) ([stringer_t](#) *chunk)

Parse an ephemeral message chunk. A public encryption key is required, while the ephemeral signing key is optional.

5.385.1 Function Documentation

5.385.1.1 [prime_ephemeral_chunk_t](#) * [ephemeral_chunk_alloc](#) (void)

[ephemeral.c](#)

Definition at line 34 of file [ephemeral.c](#).

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_ephemeral_chunk_t](#).

Referenced by [ephemeral_chunk_get\(\)](#), and [ephemeral_chunk_set\(\)](#).

5.385.1.2 [stringer_t](#) * [ephemeral_chunk_buffer](#) ([prime_ephemeral_chunk_t](#) * *chunk*)

Definition at line 49 of file [ephemeral.c](#).

Referenced by [naked_message_set\(\)](#).

5.385.1.3 void [ephemeral_chunk_cleanup](#) ([prime_ephemeral_chunk_t](#) * *chunk*)

Definition at line 27 of file [ephemeral.c](#).

References [ephemeral_chunk_free\(\)](#).

Referenced by [naked_message_get\(\)](#).

5.385.1.4 void [ephemeral_chunk_free](#) ([prime_ephemeral_chunk_t](#) * *chunk*)

Definition at line 10 of file [ephemeral.c](#).

References [ed25519_free\(\)](#), [log_pedantic](#), [mm_free\(\)](#), [secp256k1_free\(\)](#), and [st_free\(\)](#).

Referenced by [encrypted_message_free\(\)](#), [ephemeral_chunk_cleanup\(\)](#), [ephemeral_chunk_get\(\)](#), [ephemeral_chunk_set\(\)](#), and [naked_message_get\(\)](#).

5.385.1.5 prime_ephemeral_chunk_t* ephemeral_chunk_get (ed25519_key_t * *signing*, secp256k1_key_t * *encryption*)

Generate an ephemeral message chunk. A public encryption key is required, while the public signing key is optional, but will be included if a signing key is provided.

Definition at line 64 of file ephemeral.c.

References chunk_header_write(), ED25519_KEY_PUB_LEN, ED25519_PRIV, ED25519_PUB, ed25519_public_get(), ed25519_public_set(), ephemeral_chunk_alloc(), ephemeral_chunk_free(), log_pedantic, MANAGEDBUF, PRIME_CHUNK_EPHEMERAL, prime_ephemeral_chunk_t, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), secp256k1_public_set(), st_length_get(), st_merge, USER_ENCRYPTION_KEY, and USER_SIGNING_KEY.

Referenced by naked_message_set().

5.385.1.6 prime_ephemeral_chunk_t* ephemeral_chunk_set (stringer_t * *chunk*)

Parse an ephemeral message chunk. A public encryption key is required, while the ephemeral signing key is optional.

Definition at line 111 of file ephemeral.c.

References chunk_header_size(), chunk_header_type(), data, ed25519_public_set(), ephemeral_chunk_alloc(), ephemeral_chunk_free(), FOREIGNDATA, HEAP, JOINTED, log_pedantic, pl_data_get(), pl_init(), PLACER, PLACER_T, placer_t, PRIME_CHUNK_EPHEMERAL, prime_ephemeral_chunk_t, secp256k1_public_set(), st_alloc_opts(), st_data_get(), st_data_set(), st_empty_out(), st_length_get(), and st_length_set().

Referenced by naked_message_get().

5.386 src/providers/prime/messages/chunks/keks.c File Reference

```
#include "magma.h"
```

Functions

- void [keks_free](#) ([prime_chunk_keks_t](#) *keks)
- void [keks_cleanup](#) ([prime_chunk_keks_t](#) *keks)
- [prime_chunk_keks_t](#) * [keks_alloc](#) (void)
keks.c
- [prime_chunk_keks_t](#) * [keks_get](#) ([prime_chunk_keys_t](#) *keys, [prime_chunk_keks_t](#) *keks)
Uses the private ephemeral encryption key to compute a KEK with all of the provided actor public keys.
- [prime_chunk_keks_t](#) * [keks_set](#) ([prime_chunk_keys_t](#) *keys, [prime_chunk_keks_t](#) *keks)
Uses the public ephemeral encryption key to compute a KEK with all of the provided actor private keys.

5.386.1 Function Documentation

5.386.1.1 [prime_chunk_keks_t](#)* [keks_alloc](#) (void)

[keks.c](#)

Definition at line 34 of file [keks.c](#).

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_chunk_keks_t](#).

Referenced by [keks_get\(\)](#), and [keks_set\(\)](#).

5.386.1.2 void [keks_cleanup](#) ([prime_chunk_keks_t](#) * *keks*)

Definition at line 25 of file [keks.c](#).

References [keks_free\(\)](#).

5.386.1.3 void [keks_free](#) ([prime_chunk_keks_t](#) * *keks*)

Definition at line 10 of file [keks.c](#).

References [log_pedantic](#), [mm_free\(\)](#), and [st_cleanup](#).

Referenced by [keks_cleanup\(\)](#), [keks_get\(\)](#), [keks_set\(\)](#), and [naked_message_get\(\)](#).

5.386.1.4 [prime_chunk_keks_t](#)* [keks_get](#) ([prime_chunk_keys_t](#) * *keys*, [prime_chunk_keks_t](#) * *keks*)

Uses the private ephemeral encryption key to compute a KEK with all of the provided actor public keys.

Definition at line 53 of file [keks.c](#).

References [keks_alloc\(\)](#), [keks_free\(\)](#), [prime_chunk_keks_t](#), [secp256k1_compute_kek\(\)](#), [SECP256K1_PRIV](#), [SECP256K1_PUB](#), and [secp256k1_type\(\)](#).

Referenced by [naked_message_set\(\)](#).

5.386.1.5 prime_chunk_keks_t* keks_set (prime_chunk_keys_t * *keys*, prime_chunk_keks_t * *keks*)

Uses the public ephemeral encryption key to compute a KEK with all of the provided actor private keys.

Definition at line 98 of file keys.c.

References keks_alloc(), keks_free(), prime_chunk_keks_t, secp256k1_compute_kek(), SECP256K1_PRIV, SECP256K1_PUB, and secp256k1_type().

Referenced by naked_message_get().

5.387 src/providers/prime/messages/chunks/signature.c File Reference

```
#include "magma.h"
```

Functions

- void [signature_tree_free](#) (prime_signature_tree_t *chunk)
- void [signature_tree_cleanup](#) (prime_signature_tree_t *chunk)
- prime_signature_tree_t * [signature_tree_alloc](#) (void)
- int_t [signature_tree_add](#) (prime_signature_tree_t *chunk, stringer_t *data)
- stringer_t * [signature_tree_get](#) (ed25519_key_t *signing, prime_signature_tree_t *chunk, prime_chunk_keks_t *keks)
Calculates the tree signature value by concatenating the hashes together, and generating an Ed25519 signature using the result.
- int_t [signature_tree_verify](#) (ed25519_key_t *signing, prime_signature_tree_t *chunk, prime_chunk_keks_t *keks, stringer_t *data)
Verify an Ed25519 signature using the EdDSA algorithm.
- stringer_t * [signature_full_get](#) (prime_message_chunk_type_t type, ed25519_key_t *signing, prime_chunk_keks_t *keks, stringer_t *data)
signature.c
- int_t [signature_full_verify](#) (ed25519_key_t *signing, prime_chunk_keks_t *keks, stringer_t *data, stringer_t *chunk)
Verify an Ed25519 signature using the EdDSA algorithm.

5.387.1 Function Documentation

5.387.1.1 stringer_t* [signature_full_get](#) (prime_message_chunk_type_t type, ed25519_key_t * signing, prime_chunk_keks_t * keks, stringer_t * data)

[signature.c](#)

Definition at line 191 of file [signature.c](#).

References [ED25519_PRIV](#), [ed25519_sign\(\)](#), [ED25519_SIGNATURE_LEN](#), [ed25519_type\(\)](#), [hash_sha512\(\)](#), [MANAGEDBUF](#), [PLACER](#), [placer_t](#), [prime_chunk_slots_t](#), [rand_write\(\)](#), [slots_buffer\(\)](#), [slots_free\(\)](#), [slots_set\(\)](#), [st_alloc\(\)](#), [st_cleanup](#), [st_length_get\(\)](#), [st_write](#), and [st_xor\(\)](#).

Referenced by [naked_message_set\(\)](#).

5.387.1.2 int_t [signature_full_verify](#) (ed25519_key_t * signing, prime_chunk_keks_t * keks, stringer_t * data, stringer_t * chunk)

Verify an Ed25519 signature using the EdDSA algorithm.

Returns:

0 for successful signature verification, -1 for a signature verification failures, -2 for processing or parameter issue.

Definition at line 234 of file [signature.c](#).

References [chunk_header_type\(\)](#), [ED25519_PRIV](#), [ED25519_PUB](#), [ED25519_SIGNATURE_LEN](#), [ed25519_type\(\)](#), [ed25519_verify\(\)](#), [hash_sha512\(\)](#), [MANAGEDBUF](#), [pl_init\(\)](#), [placer_t](#), [PRIME_SIGNATURE_USER](#), [SECP256K1_SHARED_SECRET_LEN](#), [slots_count\(\)](#), [slots_get\(\)](#), [st_data_get\(\)](#), [st_length_get\(\)](#), [st_xor\(\)](#), and [type\(\)](#).

Referenced by [naked_message_get\(\)](#).

5.387.1.3 int_t signature_tree_add (prime_signature_tree_t * chunk, stringer_t * data)

Definition at line 52 of file signature.c.

References hash_sha512(), inx_count(), inx_insert(), M_TYPE_UINT64, st_free(), st_length_get(), multi_t::type, multi_t::u64, and multi_t::val.

Referenced by naked_message_get(), and naked_message_set().

5.387.1.4 prime_signature_tree_t* signature_tree_alloc (void)

Definition at line 32 of file signature.c.

References inx_alloc(), log_pedantic, M_INX_LINKED, mm_alloc(), mm_wipe(), prime_signature_tree_t, and st_free().

Referenced by naked_message_get(), and naked_message_set().

5.387.1.5 void signature_tree_cleanup (prime_signature_tree_t * chunk)

Definition at line 25 of file signature.c.

References signature_tree_free().

5.387.1.6 void signature_tree_free (prime_signature_tree_t * chunk)

Definition at line 10 of file signature.c.

References inx_free(), log_pedantic, and mm_free().

Referenced by naked_message_get(), naked_message_set(), and signature_tree_cleanup().

5.387.1.7 stringer_t* signature_tree_get (ed25519_key_t * signing, prime_signature_tree_t * chunk, prime_chunk_keks_t * keks)

Calculates the tree signature value by concatenating the hashes together, and generating an Ed25519 signature using the result.

Definition at line 79 of file signature.c.

References count, ED25519_PRIV, ed25519_sign(), ed25519_type(), hash_sha512(), HEAP, inx_count(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), JOINTED, MANAGED_T, MANAGEDBUF, PLACER, placer_t, prime_chunk_slots_t, PRIME_SIGNATURE_TREE, rand_write(), slots_buffer(), slots_free(), slots_set(), st_alloc(), st_alloc_opts(), st_append, st_cleanup, st_free(), st_length_get(), st_write, st_xor(), and type().

Referenced by naked_message_set().

5.387.1.8 int_t signature_tree_verify (ed25519_key_t * signing, prime_signature_tree_t * chunk, prime_chunk_keks_t * keks, stringer_t * data)

Verify an Ed25519 signature using the EdDSA algorithm.

Returns:

0 for successful signature verification, -1 for a signature verification failures, -2 for processing or parameter issue.

Definition at line 146 of file signature.c.

References chunk_header_type(), count, ED25519_PRIV, ED25519_PUB, ED25519_SIGNATURE_LEN, ed25519_type(), ed25519_verify(), hash_sha512(), HEAP, inx_count(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), JOINTED, MANAGED_T, MANAGEDBUF, pl_init(), placer_t, PRIME_SIGNATURE_TREE, SECP256K1_SHARED_SECRET_LEN, slots_count(), slots_get(), st_alloc_opts(), st_append, st_data_get(), st_free(), st_length_get(), and st_xor().

Referenced by naked_message_get().

5.388 src/providers/prime/messages/chunks/slots.c File Reference

```
#include "magma.h"
```

Defines

- #define [PRIME_ACTOR_NONE](#) 0x00
- #define [PRIME_ACTOR_AUTHOR](#) 0x01
- #define [PRIME_ACTOR_ORIGIN](#) 0x02
- #define [PRIME_ACTOR_DESTINATION](#) 0x04
- #define [PRIME_ACTOR_RECIPIENT](#) 0x08

Functions

- void [slots_free](#) ([prime_chunk_slots_t](#) *slots)
- void [slots_cleanup](#) ([prime_chunk_slots_t](#) *slots)
- [prime_chunk_slots_t](#) * [slots_alloc](#) ([prime_message_chunk_type_t](#) type)
- [int_t](#) [slots_actors](#) ([prime_message_chunk_type_t](#) type)
Returns a set of bitwise flags indicating which actors are allowed to access a given chunk type.
- [int_t](#) [slots_count](#) ([prime_message_chunk_type_t](#) type)
Returns the number of slots for a given chunk type.
- [placer_t](#) [slots_buffer](#) ([prime_chunk_slots_t](#) *slots)
- [stringer_t](#) * [slots_key](#) ([prime_chunk_slots_t](#) *slots, [prime_chunk_keks_t](#) *keks, [stringer_t](#) *output)
Get the unstretched key value using the first available kek with a matching keyslot value.
- [prime_chunk_slots_t](#) * [slots_set](#) ([prime_message_chunk_type_t](#) type, [stringer_t](#) *key, [prime_chunk_keks_t](#) *keks)
- [stringer_t](#) * [slots_get](#) ([prime_message_chunk_type_t](#) type, [stringer_t](#) *slots, [prime_chunk_keks_t](#) *keks, [stringer_t](#) *output)

5.388.1 Define Documentation

5.388.1.1 #define [PRIME_ACTOR_AUTHOR](#) 0x01

Definition at line 11 of file slots.c.

Referenced by [slots_actors\(\)](#), [slots_alloc\(\)](#), [slots_get\(\)](#), and [slots_set\(\)](#).

5.388.1.2 #define [PRIME_ACTOR_DESTINATION](#) 0x04

Definition at line 13 of file slots.c.

Referenced by [slots_actors\(\)](#), [slots_alloc\(\)](#), [slots_get\(\)](#), and [slots_set\(\)](#).

5.388.1.3 #define [PRIME_ACTOR_NONE](#) 0x00

Definition at line 10 of file slots.c.

Referenced by [slots_actors\(\)](#), [slots_alloc\(\)](#), [slots_get\(\)](#), and [slots_set\(\)](#).

5.388.1.4 `#define PRIME_ACTOR_ORIGIN 0x02`

Definition at line 12 of file slots.c.

Referenced by slots_actors(), slots_alloc(), slots_get(), and slots_set().

5.388.1.5 `#define PRIME_ACTOR_RECIPIENT 0x08`

Definition at line 14 of file slots.c.

Referenced by slots_actors(), slots_alloc(), slots_get(), and slots_set().

5.388.2 Function Documentation

5.388.2.1 `int_t slots_actors (prime_message_chunk_type_t type)`

Returns a set of bitwise flags indicating which actors are allowed to access a given chunk type. [slots.c](#)

Definition at line 99 of file slots.c.

References log_pedantic, PRIME_ACTOR_AUTHOR, PRIME_ACTOR_DESTINATION, PRIME_ACTOR_NONE, PRIME_ACTOR_ORIGIN, PRIME_ACTOR_RECIPIENT, PRIME_CHUNK_BODY, PRIME_CHUNK_COMMON, PRIME_CHUNK_DESTINATION, PRIME_CHUNK_HEADERS, PRIME_CHUNK_ORIGIN, PRIME_SIGNATURE_DESTINATION, PRIME_SIGNATURE_ORGIN, PRIME_SIGNATURE_TREE, and PRIME_SIGNATURE_USER.

Referenced by slots_alloc(), slots_count(), slots_get(), and slots_set().

5.388.2.2 `prime_chunk_slots_t* slots_alloc (prime_message_chunk_type_t type)`

Definition at line 39 of file slots.c.

References count, log_pedantic, mm_alloc(), mm_wipe(), pl_init(), PRIME_ACTOR_AUTHOR, PRIME_ACTOR_DESTINATION, PRIME_ACTOR_NONE, PRIME_ACTOR_ORIGIN, PRIME_ACTOR_RECIPIENT, prime_chunk_slots_t, SECP256K1_SHARED_SECRET_LEN, slots_actors(), slots_count(), and slots_free().

Referenced by slots_get(), and slots_set().

5.388.2.3 `placer_t slots_buffer (prime_chunk_slots_t * slots)`

Definition at line 156 of file slots.c.

References pl_length_get(), pl_null(), and placer_t.

Referenced by signature_full_get(), and signature_tree_get().

5.388.2.4 `void slots_cleanup (prime_chunk_slots_t * slots)`

Definition at line 30 of file slots.c.

References slots_free().

5.388.2.5 `int_t slots_count (prime_message_chunk_type_t type)`

Returns the number of slots for a given chunk type.

Definition at line 152 of file slots.c.

References bitwise_count(), and slots_actors().

Referenced by `chunk_buffer_read()`, `chunk_buffer_size()`, `chunk_header_read()`, `encrypted_chunk_get()`, `signature_full_verify()`, `signature_tree_verify()`, `slots_alloc()`, `slots_get()`, and `slots_set()`.

5.388.2.6 `void slots_free (prime_chunk_slots_t * slots)`

Definition at line 16 of file `slots.c`.

References `log_pedantic`, and `mm_free()`.

Referenced by `encrypted_chunk_free()`, `encrypted_chunk_set()`, `signature_full_get()`, `signature_tree_get()`, `slots_alloc()`, `slots_cleanup()`, `slots_get()`, and `slots_set()`.

5.388.2.7 `stringer_t* slots_get (prime_message_chunk_type_t type, stringer_t * slots, prime_chunk_keks_t * keks, stringer_t * output)`

Definition at line 273 of file `slots.c`.

References `count`, `log_pedantic`, `pl_init()`, `PRIME_ACTOR_AUTHOR`, `PRIME_ACTOR_DESTINATION`, `PRIME_ACTOR_NONE`, `PRIME_ACTOR_ORIGIN`, `PRIME_ACTOR_RECIPIENT`, `prime_chunk_slots_t`, `SECP256K1_SHARED_SECRET_LEN`, `slots_actors()`, `slots_alloc()`, `slots_count()`, `slots_free()`, `slots_key()`, `st_data_get()`, and `st_length_get()`.

Referenced by `encrypted_chunk_get()`, `signature_full_verify()`, and `signature_tree_verify()`.

5.388.2.8 `stringer_t* slots_key (prime_chunk_slots_t * slots, prime_chunk_keks_t * keks, stringer_t * output)`

Get the unstretched key value using the first available kek with a matching keyslot value.

Definition at line 170 of file `slots.c`.

References `log_pedantic`, `MANAGEDBUF`, `pl_empty()`, `SECP256K1_SHARED_SECRET_LEN`, `st_cmp_cs_eq()`, `st_free()`, `st_length_get()`, `st_output()`, `st_set()`, and `st_xor()`.

Referenced by `slots_get()`.

5.388.2.9 `prime_chunk_slots_t* slots_set (prime_message_chunk_type_t type, stringer_t * key, prime_chunk_keks_t * keks)`

Definition at line 237 of file `slots.c`.

References `PRIME_ACTOR_AUTHOR`, `PRIME_ACTOR_DESTINATION`, `PRIME_ACTOR_NONE`, `PRIME_ACTOR_ORIGIN`, `PRIME_ACTOR_RECIPIENT`, `prime_chunk_slots_t`, `slots_actors()`, `slots_alloc()`, `slots_count()`, `slots_free()`, `st_length_get()`, and `st_xor()`.

Referenced by `encrypted_chunk_set()`, `signature_full_get()`, and `signature_tree_get()`.

5.389 src/providers/prime/messages/parts/parts.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * part_buffer](#) ([prime_encrypted_chunk_t * chunk](#))
Serializes a body part, which may be made up of multiple encrypted chunks into a single serialized buffer.
- [prime_encrypted_chunk_t * part_encrypt](#) ([prime_message_chunk_type_t type](#), [ed25519_key_t * signing](#), [prime_chunk_keks_t * keks](#), [stringer_t * payload](#))
- [stringer_t * part_decrypt](#) ([ed25519_key_t * signing](#), [prime_chunk_keks_t * keks](#), [stringer_t * part](#), [stringer_t * output](#), [size_t * consumed](#))

5.389.1 Detailed Description

Definition in file [parts.c](#).

5.389.2 Function Documentation

5.389.2.1 [stringer_t * part_buffer](#) ([prime_encrypted_chunk_t * chunk](#))

Serializes a body part, which may be made up of multiple encrypted chunks into a single serialized buffer. [parts.c](#)

Definition at line 13 of file [parts.c](#).

References [encrypted_chunk_buffer\(\)](#), [st_append](#), and [st_cleanup](#).

Referenced by [naked_message_set\(\)](#).

5.389.2.2 [stringer_t * part_decrypt](#) ([ed25519_key_t * signing](#), [prime_chunk_keks_t * keks](#), [stringer_t * part](#), [stringer_t * output](#), [size_t * consumed](#))

LOW: Currently the only valid spanning chunk is the body. However, that will change when we finish implementing the spec.

Definition at line 94 of file [parts.c](#).

References [chunk_buffer_read\(\)](#), [data](#), [ED25519_PUB](#), [ed25519_type\(\)](#), [encrypted_chunk_get\(\)](#), [HEAP](#), [JOINTED](#), [log_pedantic](#), [MANAGED_T](#), [pl_null\(\)](#), [PLACER](#), [placer_t](#), [PRIME_CHUNK_BODY](#), [st_alloc_opts\(\)](#), [st_append](#), [st_cleanup](#), [st_empty_out\(\)](#), [st_free\(\)](#), and [type\(\)](#).

Referenced by [naked_message_get\(\)](#).

5.389.2.3 [prime_encrypted_chunk_t * part_encrypt](#) ([prime_message_chunk_type_t type](#), [ed25519_key_t * signing](#), [prime_chunk_keks_t * keks](#), [stringer_t * payload](#))

Definition at line 47 of file [parts.c](#).

References [data](#), [ED25519_PRIV](#), [ed25519_type\(\)](#), [encrypted_chunk_cleanup\(\)](#), [encrypted_chunk_set\(\)](#), [log_pedantic](#), [PLACER](#), [PRIME_CHUNK_FLAG_NONE](#), [PRIME_CHUNK_FLAG_SPANNING](#), [prime_encrypted_chunk_t](#), and [st_empty_out\(\)](#).

Referenced by [naked_message_set\(\)](#).

5.390 src/providers/prime/messages/parts/parts.h File Reference

DESCRIPTIONxxxGOESxxxHERE.

Functions

- [stringer_t * part_buffer](#) ([prime_encrypted_chunk_t * chunk](#))
parts.c
- [stringer_t * part_decrypt](#) ([ed25519_key_t * signing](#), [prime_chunk_keks_t * keks](#), [stringer_t * part](#), [stringer_t * output](#), [size_t * consumed](#))
- [prime_encrypted_chunk_t * part_encrypt](#) ([prime_message_chunk_type_t type](#), [ed25519_key_t * signing](#), [prime_chunk_keks_t * keks](#), [stringer_t * payload](#))

5.390.1 Detailed Description

DESCRIPTIONxxxGOESxxxHERE.

Definition in file [parts.h](#).

5.390.2 Function Documentation

5.390.2.1 [stringer_t * part_buffer](#) ([prime_encrypted_chunk_t * chunk](#))

[parts.c](#)

Definition at line 13 of file [parts.c](#).

References [encrypted_chunk_buffer\(\)](#), [st_append](#), and [st_cleanup](#).

Referenced by [naked_message_set\(\)](#).

5.390.2.2 [stringer_t * part_decrypt](#) ([ed25519_key_t * signing](#), [prime_chunk_keks_t * keks](#), [stringer_t * part](#), [stringer_t * output](#), [size_t * consumed](#))

LOW: Currently the only valid spanning chunk is the body. However, that will change when we finish implementing the spec.

Definition at line 94 of file [parts.c](#).

References [chunk_buffer_read\(\)](#), [data](#), [ED25519_PUB](#), [ed25519_type\(\)](#), [encrypted_chunk_get\(\)](#), [HEAP](#), [JOINTED](#), [log_pedantic](#), [MANAGED_T](#), [pl_null\(\)](#), [PLACER](#), [placer_t](#), [PRIME_CHUNK_BODY](#), [st_alloc_opts\(\)](#), [st_append](#), [st_cleanup](#), [st_empty_out\(\)](#), [st_free\(\)](#), and [type\(\)](#).

Referenced by [naked_message_get\(\)](#).

5.390.2.3 [prime_encrypted_chunk_t * part_encrypt](#) ([prime_message_chunk_type_t type](#), [ed25519_key_t * signing](#), [prime_chunk_keks_t * keks](#), [stringer_t * payload](#))

Definition at line 47 of file [parts.c](#).

References [data](#), [ED25519_PRIV](#), [ed25519_type\(\)](#), [encrypted_chunk_cleanup\(\)](#), [encrypted_chunk_set\(\)](#), [log_pedantic](#), [PLACER](#), [PRIME_CHUNK_FLAG_NONE](#), [PRIME_CHUNK_FLAG_SPANNING](#), [prime_encrypted_chunk_t](#), and [st_empty_out\(\)](#).

Referenced by [naked_message_set\(\)](#).

5.391 src/providers/prime/prime.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t prime_start` (void)
Initialize the global PRIME structures.
- `void prime_stop` (void)
Destroy the global PRIME structures.
- `void prime_free` (prime_t *object)
- `void prime_cleanup` (prime_t *object)
- `prime_t * prime_alloc` (prime_type_t type, prime_flags_t flags)
prime.c
- `stringer_t * prime_get` (prime_t *object, prime_encoding_t encoding, stringer_t *output)
Serializes a PRIME object and returns it in binary form, or with an ASCII armor encoding.
- `prime_t * prime_set` (stringer_t *object, prime_encoding_t encoding, prime_flags_t flags)
Parses a serialized PRIME object into a working context.
- `prime_t * prime_key_generate` (prime_artifact_type_t type, prime_flags_t flags)
Generates a new organizational or user key.
- `prime_t * prime_signet_generate` (prime_t *object)
Takes an organizational key, and generates the corresponding signet.
- `stringer_t * prime_signet_fingerprint` (prime_t *object, stringer_t *output)
Takes an organizational signet, or a user signet, and returns the corresponding cryptographic fingerprint.
- `bool_t prime_signet_validate` (prime_t *object, prime_t *validator)
Takes a signet, and validates the self signature. If the object is a user signet, and the validator is a user signet, then the custody signature is verified. If the validator is an organizational signet, then the organizational signature is validated. If the object is a signing request, then the validator must be a user signet, and the chain of custody signature is verified.
- `prime_t * prime_request_generate` (prime_t *object, prime_t *previous)
Takes a user key, and possibly the previous user key, and generate a signet signing request.
- `prime_t * prime_request_sign` (prime_t *request, prime_t *org)
Takes a user signing request, and an organizational key, and returns a signed user signet.
- `stringer_t * prime_message_encrypt` (stringer_t *message, prime_t *author, prime_t *origin, prime_t *destination, prime_t *recipient)
Encrypt a message.
- `stringer_t * prime_message_decrypt` (stringer_t *message, prime_t *org, prime_t *user)
Decrypt a message.
- `stringer_t * prime_key_encrypt` (stringer_t *key, prime_t *object, prime_encoding_t encoding, stringer_t *output)
Encrypt an organizational or user key using a STACIE realm key.
- `prime_t * prime_key_decrypt` (stringer_t *key, stringer_t *object, prime_encoding_t encoding, prime_flags_t flags)
Decrypt an organizational or user key using a STACIE realm key.

Variables

- `prime_t * org_key` = NULL
- `prime_t * org_signet` = NULL
- `EC_GROUP * prime_curve_group` = NULL

5.391.1 Function Documentation

5.391.1.1 `prime_t* prime_alloc (prime_type_t type, prime_flags_t flags)`

[prime.c](#)

Definition at line 155 of file `prime.c`.

References `log_pedantic`, `mm_alloc()`, `mm_wipe()`, `prime_free()`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_MESSAGE_NAKED`, `PRIME_ORG_KEY`, `PRIME_ORG_SIGNET`, `prime_t`, `PRIME_USER_KEY`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_key_decrypt()`, `prime_key_generate()`, `prime_message_encrypt()`, `prime_request_generate()`, `prime_request_sign()`, `prime_set()`, and `prime_signet_generate()`.

5.391.1.2 `void prime_cleanup (prime_t * object)`

Definition at line 148 of file `prime.c`.

References `prime_free()`.

Referenced by `mail_store_message()`, `meta_crypto_keys_create()`, `meta_free()`, and `smtp_free_inbound()`.

5.391.1.3 `void prime_free (prime_t * object)`

Definition at line 99 of file `prime.c`.

References `encrypted_message_free()`, `log_pedantic`, `mm_free()`, `org_key_free()`, `org_signet_free()`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_MESSAGE_NAKED`, `PRIME_ORG_KEY`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, `user_key_free()`, and `user_signet_free()`.

Referenced by `meta_crypto_keys_create()`, `prime_alloc()`, `prime_cleanup()`, `prime_key_decrypt()`, `prime_message_decrypt()`, `prime_message_encrypt()`, `prime_request_generate()`, `prime_request_sign()`, `prime_set()`, `prime_signet_generate()`, and `prime_stop()`.

5.391.1.4 `stringer_t* prime_get (prime_t * object, prime_encoding_t encoding, stringer_t * output)`

Serializes a PRIME object and returns it in binary form, or with an ASCII armor encoding.

Definition at line 205 of file `prime.c`.

References `ARMORED`, `log_pedantic`, `org_key_get()`, `org_signet_get()`, `PRIME_ORG_KEY`, `PRIME_ORG_SIGNET`, `prime_pem_wrap()`, `PRIME_USER_KEY`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, `st_dup()`, `st_free()`, `user_key_get()`, `user_request_get()`, and `user_signet_get()`.

Referenced by `meta_crypto_keys_create()`.

5.391.1.5 `prime_t* prime_key_decrypt (stringer_t * key, stringer_t * object, prime_encoding_t encoding, prime_flags_t flags)`

Decrypt an organizational or user key using a STACIE realm key.

Definition at line 659 of file `prime.c`.

References ARMORED, log_pedantic, org_encrypted_key_set(), prime_alloc(), prime_free(), prime_header_read(), PRIME_ORG_KEY, PRIME_ORG_KEY_ENCRYPTED, prime_pem_unwrap(), prime_t, PRIME_USER_KEY, PRIME_USER_KEY_ENCRYPTED, st_cleanup, st_length_get(), STACIE_KEY_LENGTH, type(), and user_encrypted_key_set().

Referenced by meta_update_keys().

5.391.1.6 stringer_t* prime_key_encrypt (stringer_t * *key*, prime_t * *object*, prime_encoding_t *encoding*, stringer_t * *output*)

Encrypt an organizational or user key using a STACIE realm key.

Definition at line 606 of file prime.c.

References ARMORED, log_pedantic, org_encrypted_key_get(), PRIME_ORG_KEY, prime_pem_wrap(), PRIME_USER_KEY, st_dupe(), st_free(), and user_encrypted_key_get().

Referenced by meta_crypto_keys_create().

5.391.1.7 prime_t* prime_key_generate (prime_artifact_type_t *type*, prime_flags_t *flags*)

Generates a new organizational or user key.

Definition at line 358 of file prime.c.

References log_pedantic, mm_free(), org_key_generate(), prime_alloc(), PRIME_ORG_KEY, prime_t, PRIME_USER_KEY, and user_key_generate().

Referenced by meta_crypto_keys_create().

5.391.1.8 stringer_t* prime_message_decrypt (stringer_t * *message*, prime_t * *org*, prime_t * *user*)

Decrypt a message.

Definition at line 586 of file prime.c.

References log_pedantic, naked_message_get(), prime_free(), PRIME_ORG_SIGNET, and PRIME_USER_KEY.

Referenced by mail_load_message().

5.391.1.9 stringer_t* prime_message_encrypt (stringer_t * *message*, prime_t * *author*, prime_t * *origin*, prime_t * *destination*, prime_t * *recipient*)

Encrypt a message.

Definition at line 558 of file prime.c.

References log_pedantic, naked_message_set(), NONE, prime_alloc(), prime_free(), PRIME_MESSAGE_NAKED, PRIME_ORG_KEY, prime_t, and PRIME_USER_SIGNET.

Referenced by mail_store_message().

5.391.1.10 prime_t* prime_request_generate (prime_t * *object*, prime_t * *previous*)

Takes a user key, and possibly the previous user key, and generate a signet signing request.

Definition at line 505 of file prime.c.

References log_pedantic, NONE, prime_alloc(), prime_free(), prime_t, PRIME_USER_KEY, PRIME_USER_SIGNING_REQUEST, user_request_generate(), and user_request_rotation().

Referenced by meta_crypto_keys_create().

5.391.1.11 prime_t* prime_request_sign (prime_t * request, prime_t * org)

Takes a user signing request, and an organizational key, and returns a signed user signet.

Definition at line 532 of file prime.c.

References log_pedantic, NONE, prime_alloc(), prime_free(), PRIME_ORG_KEY, prime_t, PRIME_USER_SIGNET, PRIME_USER_SIGNING_REQUEST, and user_request_sign().

Referenced by meta_crypto_keys_create().

5.391.1.12 prime_t* prime_set (stringer_t * object, prime_encoding_t encoding, prime_flags_t flags)

Parses a serialized PRIME object into a working context.

Definition at line 267 of file prime.c.

References ARMORED, log_pedantic, org_key_set(), org_signet_set(), prime_alloc(), prime_free(), prime_header_read(), PRIME_ORG_KEY, PRIME_ORG_SIGNET, prime_pem_unwrap(), prime_t, PRIME_USER_KEY, PRIME_USER_SIGNET, PRIME_USER_SIGNING_REQUEST, st_cleanup, type(), user_key_set(), user_request_set(), and user_signet_set().

Referenced by meta_update_keys(), prime_start(), and smtp_fetch_inbound().

5.391.1.13 stringer_t* prime_signet_fingerprint (prime_t * object, stringer_t * output)

Takes an organizational signet, or a user signet, and returns the corresponding cryptographic fingerprint.

Definition at line 419 of file prime.c.

References log_pedantic, org_signet_fingerprint(), PRIME_ORG_SIGNET, PRIME_USER_SIGNET, and user_signet_fingerprint().

5.391.1.14 prime_t* prime_signet_generate (prime_t * object)

Takes an organizational key, and generates the corresponding signet.

Definition at line 395 of file prime.c.

References log_pedantic, NONE, org_signet_generate(), prime_alloc(), prime_free(), PRIME_ORG_KEY, PRIME_ORG_SIGNET, and prime_t.

5.391.1.15 bool_t prime_signet_validate (prime_t * object, prime_t * validator)

Takes a signet, and validates the self signature. If the object is a user signet, and the validator is a user signet, then the custody signature is verified. If the validator is an organizational signet, then the organizational signature is validated. If the object is a signing request, then the validator must be a user signet, and the chain of custody signature is verified.

Definition at line 450 of file prime.c.

References log_pedantic, org_signet_verify(), PRIME_ORG_SIGNET, PRIME_USER_SIGNET, PRIME_USER_SIGNING_REQUEST, user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.391.1.16 bool_t prime_start (void)

Initialize the global PRIME structures.

Returns:

returns true if everything initializes properly, or false if an error occurs.

TODO: Verify that the provided signet matches the provided key file.

Definition at line 19 of file prime.c.

References ARMORED, magma_t::dime, EC_GROUP_free_d, EC_GROUP_new_by_curve_name_d, EC_GROUP_precompute_mult_d, EC_GROUP_set_point_conversion_form_d, file_load(), file_world_accessible(), magma_t::key, log_critical, log_error, magma, MEMORY_BUF, NONE, org_key, org_signet, prime_curve_group, prime_set(), SECURE, magma_t::signet, ssl_error_string(), st_char_get(), st_free(), and st_length_int().

Referenced by process_start().

5.391.1.17 void prime_stop (void)

Destroy the global PRIME structures.

Returns:

This function returns no value.

Definition at line 83 of file prime.c.

References EC_GROUP_free_d, org_key, org_signet, prime_curve_group, and prime_free().

Referenced by process_stop().

5.391.2 Variable Documentation

5.391.2.1 prime_t* org_key = NULL

Definition at line 10 of file prime.c.

Referenced by mail_store_message(), meta_crypto_keys_create(), prime_start(), and prime_stop().

5.391.2.2 prime_t* org_signet = NULL

Definition at line 11 of file prime.c.

Referenced by _get_signet(), mail_load_message(), prime_start(), and prime_stop().

5.391.2.3 EC_GROUP* prime_curve_group = NULL

Definition at line 12 of file prime.c.

Referenced by prime_start(), prime_stop(), and secp256k1_alloc().

5.392 src/providers/prime/prime.h File Reference

```
#include "transposition/transposition.h"
#include "cryptography/cryptography.h"
#include "messages/messages.h"
#include "signets/signets.h"
#include "keys/keys.h"
```

Defines

- #define [SECP256K1_KEY_PUB_LEN](#) 33
- #define [SECP256K1_KEY_PRIV_LEN](#) 32
- #define [SECP256K1_SHARED_SECRET_LEN](#) 32
- #define [ED25519_KEY_PUB_LEN](#) 32
- #define [ED25519_KEY_PRIV_LEN](#) 32
- #define [ED25519_SIGNATURE_LEN](#) 64
- #define [AES_TAG_LEN](#) 16
- #define [AES_KEY_LEN](#) 32
- #define [AES_BLOCK_LEN](#) 16
- #define [AES_VECTOR_LEN](#) 16

Typedefs

- typedef uint16_t [prime_type_t](#)
- typedef void [secp256k1_key_t](#)

Enumerations

- enum [ed25519_key_type_t](#) { [ED25519_ERR](#), [ED25519_PUB](#), [ED25519_PRIV](#) }
- enum [secp256k1_key_type_t](#) { [SECP256K1_ERR](#), [SECP256K1_PUB](#), [SECP256K1_PRIV](#) }
- enum [prime_encoding_t](#) { [BINARY](#), [ARMORED](#), [DEBUG](#) }
- enum [prime_flags_t](#) { [NONE](#), [SECURITY](#) }
- enum [prime_message_type_t](#) {
[PRIME_MESSAGE_ENCRYPTED](#) = 1847, [PRIME_MESSAGE_SENT](#) = 1851, [PRIME_MESSAGE_DRAFT](#) = 1861, [PRIME_MESSAGE_NAKED](#) = 1908,
[PRIME_MESSAGE_BOUNCE](#) = 1931, [PRIME_MESSAGE_FORWARD](#) = 1948, [PRIME_MESSAGE_ABUSE](#) = 2001, [PRIME_BINARY_OBJECT](#) = 1837,
[PRIME_PROTOCOL_TICKET](#) = 1841 }
- enum [prime_artifact_type_t](#) {
[PRIME_ORG_SIGNET](#) = 1776, [PRIME_ORG_KEY](#) = 1952, [PRIME_ORG_KEY_ENCRYPTED](#) = 1947, [PRIME_USER_SIGNING_REQUEST](#) = 1215,
[PRIME_USER_SIGNET](#) = 1789, [PRIME_USER_KEY](#) = 2013, [PRIME_USER_KEY_ENCRYPTED](#) = 1976 }
- enum [prime_message_chunk_type_t](#) {
[PRIME_CHUNK_INVALID](#) = -1, [PRIME_CHUNK_TRACING](#) = 0, [PRIME_CHUNK_EPHEMERAL](#) = 1, [PRIME_CHUNK_ORIGIN](#) = 2,
[PRIME_CHUNK_DESTINATION](#) = 3, [PRIME_CHUNK_COMMON](#) = 32, [PRIME_CHUNK_HEADERS](#) = 33, [PRIME_CHUNK_BODY](#) = 48,
[PRIME_SIGNATURE_TREE](#) = 224, [PRIME_SIGNATURE_USER](#) = 225, [PRIME_SIGNATURE_ORGIN](#) = 254, [PRIME_SIGNATURE_DESTINATION](#) = 255 }

- enum `prime_message_chunk_flags_t` {
`PRIME_CHUNK_FLAG_INVALID` = -1, `PRIME_CHUNK_FLAG_NONE` = 0, `PRIME_CHUNK_FLAG_ALTERNATE_PADDING` = 1, `PRIME_CHUNK_FLAG_ALTERNATE_ENCRYPT` = 2,
`PRIME_CHUNK_FLAG_COMPRESSED` = 4, `PRIME_CHUNK_FLAG_SPANNING` = 128 }
- enum `prime_org_artifact_fields_t` {
`ORG_PRIMARY_KEY` = 1, `ORG_SECONDARY_KEY` = 2, `ORG_ENCRYPTION_KEY` = 3, `ORG_SELF_SIGNATURE` = 4,
`ORG_FULL_SIGNATURE` = 253, `ORG_IDENTIFIER` = 254, `ORG_IDENTIFIABLE_SIGNATURE` = 255 }
- enum `prime_user_artifact_fields_t` {
`USER_SIGNING_KEY` = 1, `USER_ENCRYPTION_KEY` = 2, `USER_ALTERNATE_ENCRYPTION_KEY` = 3, `USER_CUSTODY_SIGNATURE` = 4,
`USER_SELF_SIGNATURE` = 5, `USER_CRYPTOSIGNATURE` = 6, `USER_FULL_SIGNATURE` = 253, `USER_IDENTIFIER` = 254,
`USER_IDENTIFIABLE_SIGNATURE` = 255 }

Functions

- struct `__attribute__((packed))`
- `prime_t * prime_alloc` (`prime_type_t` type, `prime_flags_t` flags)
prime.c
- void `prime_cleanup` (`prime_t` *object)
- void `prime_free` (`prime_t` *object)
- `stringer_t * prime_get` (`prime_t` *object, `prime_encoding_t` encoding, `stringer_t` *output)
Serializes a PRIME object and returns it in binary form, or with an ASCII armor encoding.
- `prime_t * prime_key_decrypt` (`stringer_t` *key, `stringer_t` *object, `prime_encoding_t` encoding, `prime_flags_t` flags)
Decrypt an organizational or user key using a STACIE realm key.
- `stringer_t * prime_key_encrypt` (`stringer_t` *key, `prime_t` *object, `prime_encoding_t` encoding, `stringer_t` *output)
Encrypt an organizational or user key using a STACIE realm key.
- `prime_t * prime_key_generate` (`prime_artifact_type_t` type, `prime_flags_t` flags)
Generates a new organizational or user key.
- `stringer_t * prime_message_decrypt` (`stringer_t` *message, `prime_t` *org, `prime_t` *user)
Decrypt a message.
- `stringer_t * prime_message_encrypt` (`stringer_t` *message, `prime_t` *author, `prime_t` *origin, `prime_t` *destination, `prime_t` *recipient)
Encrypt a message.
- `prime_t * prime_request_generate` (`prime_t` *object, `prime_t` *previous)
Takes a user key, and possibly the previous user key, and generate a signet signing request.
- `prime_t * prime_request_sign` (`prime_t` *request, `prime_t` *org)
Takes a user signing request, and an organizational key, and returns a signed user signet.
- `prime_t * prime_set` (`stringer_t` *object, `prime_encoding_t` encoding, `prime_flags_t` flags)
Parses a serialized PRIME object into a working context.
- `stringer_t * prime_signet_fingerprint` (`prime_t` *object, `stringer_t` *output)
Takes an organizational signet, or a user signet, and returns the corresponding cryptographic fingerprint.

- `prime_t * prime_signet_generate (prime_t *object)`

Takes an organizational key, and generates the corresponding signet.

- `bool_t prime_signet_validate (prime_t *object, prime_t *validator)`

Takes a signet, and validates the self signature. If the object is a user signet, and the validator is a user signet, then the custody signature is verified. If the validator is an organizational signet, then the organizational signature is validated. If the object is a signing request, then the validator must be a user signet, and the chain of custody signature is verified.

- `bool_t prime_start (void)`

Initialize the global PRIME structures.

- `void prime_stop (void)`

Destroy the global PRIME structures.

Variables

- `ed25519_key_t`
- `prime_user_key_t`
- `prime_org_key_t`
- `prime_user_signet_t`
- `prime_org_signet_t`
- `prime_chunk_keys_t`
- `prime_chunk_keks_t`
- `prime_chunk_slots_t`
- `prime_ephemeral_chunk_t`
- `prime_encrypted_chunk_t`
- `prime_signature_tree_t`
- `prime_message_t`
- `prime_t`
- `prime_t * org_key`
- `prime_t * org_signet`

5.392.1 Define Documentation

5.392.1.1 #define AES_BLOCK_LEN 16

Definition at line 21 of file prime.h.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, and `aes_chunk_encrypt()`.

5.392.1.2 #define AES_KEY_LEN 32

Definition at line 20 of file prime.h.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, and `aes_cipher_key()`.

5.392.1.3 #define AES_TAG_LEN 16

Definition at line 19 of file prime.h.

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `aes_cipher_key()`, and `aes_tag_shard()`.

5.392.1.4 #define AES_VECTOR_LEN 16

Definition at line 22 of file prime.h.

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), aes_cipher_key(), aes_tag_shard(), and aes_vector_shard().

5.392.1.5 #define ED25519_KEY_PRIV_LEN 32

Definition at line 16 of file prime.h.

Referenced by ed25519_generate(), ed25519_private_get(), ed25519_private_set(), org_key_get(), and user_key_get().

5.392.1.6 #define ED25519_KEY_PUB_LEN 32

Definition at line 15 of file prime.h.

Referenced by ed25519_generate(), ed25519_public_get(), ed25519_public_set(), ephemeral_chunk_get(), org_signet_fingerprint(), org_signet_generate(), org_signet_get(), org_signet_verify(), user_request_generate(), user_request_get(), user_request_rotation(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_get(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.392.1.7 #define ED25519_SIGNATURE_LEN 64

Definition at line 17 of file prime.h.

Referenced by ed25519_sign(), ed25519_verify(), encrypted_chunk_get(), org_signet_fingerprint(), org_signet_get(), prime_unpack_fields(), prime_unpack_validate(), signature_full_get(), signature_full_verify(), signature_tree_verify(), user_request_get(), user_request_rotation(), user_request_sign(), user_request_verify_self(), user_signet_fingerprint(), user_signet_get(), user_signet_verify_org(), and user_signet_verify_self().

5.392.1.8 #define SECP256K1_KEY_PRIV_LEN 32

Definition at line 12 of file prime.h.

Referenced by org_key_get(), secp256k1_private_get(), secp256k1_private_set(), and user_key_get().

5.392.1.9 #define SECP256K1_KEY_PUB_LEN 33

Definition at line 11 of file prime.h.

Referenced by ephemeral_chunk_get(), org_signet_fingerprint(), org_signet_generate(), org_signet_get(), org_signet_verify(), secp256k1_public_get(), secp256k1_public_set(), user_request_generate(), user_request_get(), user_request_rotation(), user_request_sign(), user_request_verify_chain_of_custody(), user_request_verify_self(), user_signet_fingerprint(), user_signet_get(), user_signet_verify_chain_of_custody(), user_signet_verify_org(), and user_signet_verify_self().

5.392.1.10 #define SECP256K1_SHARED_SECRET_LEN 32

Definition at line 13 of file prime.h.

Referenced by chunk_buffer_read(), chunk_buffer_size(), chunk_header_read(), encrypted_chunk_get(), secp256k1_compute_kek(), signature_full_verify(), signature_tree_verify(), slots_alloc(), slots_get(), and slots_key().

5.392.2 Typedef Documentation

5.392.2.1 prime_type_t

Definition at line 167 of file prime.h.

5.392.2.2 typedef void secp256k1_key_t

Definition at line 173 of file prime.h.

5.392.3 Enumeration Type Documentation

5.392.3.1 enum ed25519_key_type_t

Enumerator:

ED25519_ERR
ED25519_PUB
ED25519_PRIV

Definition at line 27 of file prime.h.

5.392.3.2 enum prime_artifact_type_t

Enumerator:

PRIME_ORG_SIGNET Organizational signet.
PRIME_ORG_KEY Organizational key.
PRIME_ORG_KEY_ENCRYPTED Encrypted organizational key.
PRIME_USER_SIGNING_REQUEST User signing request.
PRIME_USER_SIGNET User signet.
PRIME_USER_KEY User key.
PRIME_USER_KEY_ENCRYPTED Encrypted user key.

Definition at line 79 of file prime.h.

5.392.3.3 enum prime_encoding_t

Enumerator:

BINARY Serialized object in binary form.
ARMORED An object that has been armored using the Privacy Enhanced Message format.
DEBUG Printable version of the object, with labels, and binary data encoded using base64.

Definition at line 45 of file prime.h.

5.392.3.4 enum prime_flags_t

Enumerator:

NONE
SECURITY Store the object in secure memory.

Definition at line 54 of file prime.h.

5.392.3.5 enum prime_message_chunk_flags_t

Enumerator:

PRIME_CHUNK_FLAG_INVALID

PRIME_CHUNK_FLAG_NONE The empty set.

PRIME_CHUNK_FLAG_ALTERNATE_PADDING Alternate chunk padding algorithm.

PRIME_CHUNK_FLAG_ALTERNATE_ENCRYPT Payload is further encrypted by the alternate cipher suite.

PRIME_CHUNK_FLAG_COMPRESSED The data was compressed before being encrypted.

PRIME_CHUNK_FLAG_SPANNING The payload continues into the next available chunk.

Definition at line 121 of file prime.h.

5.392.3.6 enum prime_message_chunk_type_t

Enumerator:

PRIME_CHUNK_INVALID

PRIME_CHUNK_TRACING Tracing data.

PRIME_CHUNK_EPHEMERAL Ephemeral chunk.

PRIME_CHUNK_ORIGIN Origin chunk.

PRIME_CHUNK_DESTINATION Destination chunk.

PRIME_CHUNK_COMMON Common headers chunk.

PRIME_CHUNK_HEADERS Remaining headers chunk.

PRIME_CHUNK_BODY Naked unstructured message body.

PRIME_SIGNATURE_TREE

PRIME_SIGNATURE_USER

PRIME_SIGNATURE_ORGIN

PRIME_SIGNATURE_DESTINATION

Definition at line 92 of file prime.h.

5.392.3.7 enum prime_message_type_t

Enumerator:

PRIME_MESSAGE_ENCRYPTED An encrypted message.

PRIME_MESSAGE_SENT An encrypted, appended, sent message.

PRIME_MESSAGE_DRAFT An encrypted, appended, message draft.

PRIME_MESSAGE_NAKED An encrypted, imported, unstructured, naked message.

PRIME_MESSAGE_BOUNCE An encapsulated, encrypted message, that has bounced.

PRIME_MESSAGE_FORWARD An encapsulated, encrypted message, that has been forwarded.

PRIME_MESSAGE_ABUSE An encapsulated, encrypted message, sent as an abuse complaint.

PRIME_BINARY_OBJECT A binary object.

PRIME_PROTOCOL_TICKET An encrypted protocol ticket.

Definition at line 62 of file prime.h.

5.392.3.8 enum prime_org_artifact_fields_t

Enumerator:

ORG_PRIMARY_KEY
ORG_SECONDARY_KEY
ORG_ENCRYPTION_KEY
ORG_SELF_SIGNATURE
ORG_FULL_SIGNATURE
ORG_IDENTIFIER
ORG_IDENTIFIABLE_SIGNATURE

Definition at line 139 of file prime.h.

5.392.3.9 enum prime_user_artifact_fields_t

Enumerator:

USER_SIGNING_KEY
USER_ENCRYPTION_KEY
USER_ALTERNATE_ENCRYPTION_KEY
USER_CUSTODY_SIGNATURE
USER_SELF_SIGNATURE
USER_CRYPTO_SIGNATURE
USER_FULL_SIGNATURE
USER_IDENTIFIER
USER_IDENTIFIABLE_SIGNATURE

Definition at line 152 of file prime.h.

5.392.3.10 enum secp256k1_key_type_t

Enumerator:

SECP256K1_ERR
SECP256K1_PUB
SECP256K1_PRIV

Definition at line 36 of file prime.h.

5.392.4 Function Documentation

5.392.4.1 struct __attribute__((packed)) [read]

< User signing key, field 1.

< User encryption key, field 2.

< User chain of custody signature, field 4.

< User signature, field 5.

< Organizational signature, field 6.

- < Organizational signing key, field 1.
- < Organizational encryption key, field 3.
- < Organizational signature, field 4.
- < Chunk type, which curenly, should always be 1.
- < Payload length. Currently, this should only be 35 or 69.
- < Chunk type, 1 through 255.
- < Payload length, must be divisible by 16 and less than $2^{24} - 1$.

Definition at line 340 of file prime.h.

References `prime_message_t`, `prime_org_key_t`, `prime_org_signet_t`, `prime_user_key_t`, `prime_user_signet_t`, `type()`, and `__attribute__((user))`.

5.392.4.2 `prime_t* prime_alloc (prime_type_t type, prime_flags_t flags)`

[prime.c](#)

Definition at line 155 of file prime.c.

References `log_pedantic`, `mm_alloc()`, `mm_wipe()`, `prime_free()`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_MESSAGE_NAKED`, `PRIME_ORG_KEY`, `PRIME_ORG_SIGNET`, `prime_t`, `PRIME_USER_KEY`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_key_decrypt()`, `prime_key_generate()`, `prime_message_encrypt()`, `prime_request_generate()`, `prime_request_sign()`, `prime_set()`, and `prime_signet_generate()`.

5.392.4.3 `void prime_cleanup (prime_t * object)`

Definition at line 148 of file prime.c.

References `prime_free()`.

Referenced by `mail_store_message()`, `meta_crypto_keys_create()`, `meta_free()`, and `smtp_free_inbound()`.

5.392.4.4 `void prime_free (prime_t * object)`

Definition at line 99 of file prime.c.

References `encrypted_message_free()`, `log_pedantic`, `mm_free()`, `org_key_free()`, `org_signet_free()`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_MESSAGE_NAKED`, `PRIME_ORG_KEY`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, `user_key_free()`, and `user_signet_free()`.

Referenced by `meta_crypto_keys_create()`, `prime_alloc()`, `prime_cleanup()`, `prime_key_decrypt()`, `prime_message_decrypt()`, `prime_message_encrypt()`, `prime_request_generate()`, `prime_request_sign()`, `prime_set()`, `prime_signet_generate()`, and `prime_stop()`.

5.392.4.5 `stringer_t* prime_get (prime_t * object, prime_encoding_t encoding, stringer_t * output)`

Serializes a PRIME object and returns it in binary form, or with an ASCII armor encoding.

Definition at line 205 of file prime.c.

References `ARMORED`, `log_pedantic`, `org_key_get()`, `org_signet_get()`, `PRIME_ORG_KEY`, `PRIME_ORG_SIGNET`, `prime_pem_wrap()`, `PRIME_USER_KEY`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, `st_dup()`, `st_free()`, `user_key_get()`, `user_request_get()`, and `user_signet_get()`.

Referenced by `meta_crypto_keys_create()`.

5.392.4.6 **prime_t* prime_key_decrypt (stringer_t * *key*, stringer_t * *object*, prime_encoding_t *encoding*, prime_flags_t *flags*)**

Decrypt an organizational or user key using a STACIE realm key.

Definition at line 659 of file prime.c.

References ARMORED, log_pedantic, org_encrypted_key_set(), prime_alloc(), prime_free(), prime_header_read(), PRIME_ORG_KEY, PRIME_ORG_KEY_ENCRYPTED, prime_pem_unwrap(), prime_t, PRIME_USER_KEY, PRIME_USER_KEY_ENCRYPTED, st_cleanup, st_length_get(), STACIE_KEY_LENGTH, type(), and user_encrypted_key_set().

Referenced by meta_update_keys().

5.392.4.7 **stringer_t* prime_key_encrypt (stringer_t * *key*, prime_t * *object*, prime_encoding_t *encoding*, stringer_t * *output*)**

Encrypt an organizational or user key using a STACIE realm key.

Definition at line 606 of file prime.c.

References ARMORED, log_pedantic, org_encrypted_key_get(), PRIME_ORG_KEY, prime_pem_wrap(), PRIME_USER_KEY, st_dup(), st_free(), and user_encrypted_key_get().

Referenced by meta_crypto_keys_create().

5.392.4.8 **prime_t* prime_key_generate (prime_artifact_type_t *type*, prime_flags_t *flags*)**

Generates a new organizational or user key.

Definition at line 358 of file prime.c.

References log_pedantic, mm_free(), org_key_generate(), prime_alloc(), PRIME_ORG_KEY, prime_t, PRIME_USER_KEY, and user_key_generate().

Referenced by meta_crypto_keys_create().

5.392.4.9 **stringer_t* prime_message_decrypt (stringer_t * *message*, prime_t * *org*, prime_t * *user*)**

Decrypt a message.

Definition at line 586 of file prime.c.

References log_pedantic, naked_message_get(), prime_free(), PRIME_ORG_SIGNET, and PRIME_USER_KEY.

Referenced by mail_load_message().

5.392.4.10 **stringer_t* prime_message_encrypt (stringer_t * *message*, prime_t * *author*, prime_t * *origin*, prime_t * *destination*, prime_t * *recipient*)**

Encrypt a message.

Definition at line 558 of file prime.c.

References log_pedantic, naked_message_set(), NONE, prime_alloc(), prime_free(), PRIME_MESSAGE_NAKED, PRIME_ORG_KEY, prime_t, and PRIME_USER_SIGNET.

Referenced by mail_store_message().

5.392.4.11 **prime_t* prime_request_generate (prime_t * *object*, prime_t * *previous*)**

Takes a user key, and possibly the previous user key, and generate a signet signing request.

Definition at line 505 of file prime.c.

References `log_pedantic`, `NONE`, `prime_alloc()`, `prime_free()`, `prime_t`, `PRIME_USER_KEY`, `PRIME_USER_SIGNING_REQUEST`, `user_request_generate()`, and `user_request_rotation()`.

Referenced by `meta_crypto_keys_create()`.

5.392.4.12 `prime_t* prime_request_sign (prime_t * request, prime_t * org)`

Takes a user signing request, and an organizational key, and returns a signed user signet.

Definition at line 532 of file `prime.c`.

References `log_pedantic`, `NONE`, `prime_alloc()`, `prime_free()`, `PRIME_ORG_KEY`, `prime_t`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, and `user_request_sign()`.

Referenced by `meta_crypto_keys_create()`.

5.392.4.13 `prime_t* prime_set (stringer_t * object, prime_encoding_t encoding, prime_flags_t flags)`

Parses a serialized PRIME object into a working context.

Definition at line 267 of file `prime.c`.

References `ARMORED`, `log_pedantic`, `org_key_set()`, `org_signet_set()`, `prime_alloc()`, `prime_free()`, `prime_header_read()`, `PRIME_ORG_KEY`, `PRIME_ORG_SIGNET`, `prime_pem_unwrap()`, `prime_t`, `PRIME_USER_KEY`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, `st_cleanup`, `type()`, `user_key_set()`, `user_request_set()`, and `user_signet_set()`.

Referenced by `meta_update_keys()`, `prime_start()`, and `smtp_fetch_inbound()`.

5.392.4.14 `stringer_t* prime_signet_fingerprint (prime_t * object, stringer_t * output)`

Takes an organizational signet, or a user signet, and returns the corresponding cryptographic fingerprint.

Definition at line 419 of file `prime.c`.

References `log_pedantic`, `org_signet_fingerprint()`, `PRIME_ORG_SIGNET`, `PRIME_USER_SIGNET`, and `user_signet_fingerprint()`.

5.392.4.15 `prime_t* prime_signet_generate (prime_t * object)`

Takes an organizational key, and generates the corresponding signet.

Definition at line 395 of file `prime.c`.

References `log_pedantic`, `NONE`, `org_signet_generate()`, `prime_alloc()`, `prime_free()`, `PRIME_ORG_KEY`, `PRIME_ORG_SIGNET`, and `prime_t`.

5.392.4.16 `bool_t prime_signet_validate (prime_t * object, prime_t * validator)`

Takes a signet, and validates the self signature. If the object is a user signet, and the validator is a user signet, then the custody signature is verified. If the validator is an organizational signet, then the organizational signature is validated. If the object is a signing request, then the validator must be a user signet, and the chain of custody signature is verified.

Definition at line 450 of file `prime.c`.

References `log_pedantic`, `org_signet_verify()`, `PRIME_ORG_SIGNET`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, `user_request_verify_chain_of_custody()`, `user_request_verify_self()`, `user_signet_verify_chain_of_custody()`, `user_signet_verify_org()`, and `user_signet_verify_self()`.

5.392.4.17 `bool_t prime_start (void)`

Initialize the global PRIME structures.

Returns:

returns true if everything initializes properly, or false if an error occurs.

TODO: Verify that the provided signet matches the provided key file.

Definition at line 19 of file prime.c.

References ARMORED, magma_t::dime, EC_GROUP_free_d, EC_GROUP_new_by_curve_name_d, EC_GROUP_precompute_mult_d, EC_GROUP_set_point_conversion_form_d, file_load(), file_world_accessible(), magma_t::key, log_critical, log_error, magma, MEMORY_BUF, NONE, org_key, org_signet, prime_curve_group, prime_set(), SECURE, magma_t::signet, ssl_error_string(), st_char_get(), st_free(), and st_length_int().

Referenced by process_start().

5.392.4.18 void prime_stop (void)

Destroy the global PRIME structures.

Returns:

This function returns no value.

Definition at line 83 of file prime.c.

References EC_GROUP_free_d, org_key, org_signet, prime_curve_group, and prime_free().

Referenced by process_stop().

5.392.5 Variable Documentation**5.392.5.1 ed25519_key_t**

Definition at line 185 of file prime.h.

Referenced by ed25519_alloc(), ed25519_generate(), ed25519_private_set(), and ed25519_public_set().

5.392.5.2 prime_t* org_key

Definition at line 10 of file prime.c.

Referenced by mail_store_message(), meta_crypto_keys_create(), prime_start(), and prime_stop().

5.392.5.3 prime_t* org_signet

Definition at line 11 of file prime.c.

Referenced by _get_signet(), mail_load_message(), prime_start(), and prime_stop().

5.392.5.4 prime_chunk_keks_t

Definition at line 248 of file prime.h.

Referenced by keks_alloc(), keks_get(), keks_set(), and naked_message_get().

5.392.5.5 prime_chunk_keys_t

Definition at line 238 of file prime.h.

Referenced by naked_message_get().

5.392.5.6 prime_chunk_slots_t

Definition at line 259 of file prime.h.

Referenced by encrypted_chunk_set(), signature_full_get(), signature_tree_get(), slots_alloc(), slots_get(), and slots_set().

5.392.5.7 prime_encrypted_chunk_t

Definition at line 295 of file prime.h.

Referenced by encrypted_chunk_alloc(), encrypted_chunk_free(), encrypted_chunk_set(), naked_message_set(), and part_encrypt().

5.392.5.8 prime_ephemeral_chunk_t

Definition at line 274 of file prime.h.

Referenced by ephemeral_chunk_alloc(), ephemeral_chunk_get(), ephemeral_chunk_set(), and naked_message_get().

5.392.5.9 prime_message_t

Definition at line 335 of file prime.h.

Referenced by __attribute__(), encrypted_message_alloc(), and naked_message_set().

5.392.5.10 prime_org_key_t

Definition at line 201 of file prime.h.

Referenced by __attribute__(), org_encrypted_key_set(), org_key_alloc(), org_key_generate(), and org_key_set().

5.392.5.11 prime_org_signet_t

Definition at line 223 of file prime.h.

Referenced by __attribute__(), org_signet_alloc(), org_signet_generate(), and org_signet_set().

5.392.5.12 prime_signature_tree_t

Definition at line 302 of file prime.h.

Referenced by naked_message_get(), naked_message_set(), and signature_tree_alloc().

5.392.5.13 prime_t

Definition at line 361 of file prime.h.

Referenced by meta_crypto_keys_create(), prime_alloc(), prime_key_decrypt(), prime_key_generate(), prime_message_encrypt(), prime_request_generate(), prime_request_sign(), prime_set(), and prime_signet_generate().

5.392.5.14 prime_user_key_t

Definition at line 193 of file prime.h.

Referenced by __attribute__(), user_encrypted_key_set(), user_key_alloc(), user_key_generate(), and user_key_set().

5.392.5.15 `prime_user_signet_t`

Definition at line 214 of file `prime.h`.

Referenced by `__attribute__()`, `user_request_generate()`, `user_request_rotation()`, `user_request_set()`, `user_request_sign()`, `user_signet_alloc()`, and `user_signet_set()`.

5.393 src/providers/prime/signets/requests.c File Reference

```
#include "magma.h"
```

Functions

- [size_t user_request_length](#) ([prime_user_signet_t](#) *user)
- [prime_user_signet_t * user_request_sign](#) ([prime_user_signet_t](#) *request, [prime_org_key_t](#) *org)
Accepts a user signet signing request and signs it using the provided org key, returning a valid signet.
- [prime_user_signet_t * user_request_generate](#) ([prime_user_key_t](#) *user)
Derive a user signet signing request from the corresponding private key structures.
- [prime_user_signet_t * user_request_rotation](#) ([prime_user_key_t](#) *user, [prime_user_key_t](#) *previous)
Generate a user signet signing rotation request using the previous, and new user private key structures.
- [stringer_t * user_request_get](#) ([prime_user_signet_t](#) *user, [stringer_t](#) *output)
- [prime_user_signet_t * user_request_set](#) ([stringer_t](#) *user)
- [bool_t user_request_verify_self](#) ([prime_user_signet_t](#) *user)
- [bool_t user_request_verify_chain_of_custody](#) ([prime_user_signet_t](#) *user, [prime_user_signet_t](#) *previous)

5.393.1 Function Documentation

5.393.1.1 [prime_user_signet_t * user_request_generate](#) ([prime_user_key_t](#) * user)

Derive a user signet signing request from the corresponding private key structures. [requests.c](#)

Definition at line 101 of file [requests.c](#).

References [ED25519_KEY_PUB_LEN](#), [ED25519_PRIV](#), [ed25519_public_get\(\)](#), [ed25519_public_set\(\)](#), [ed25519_sign\(\)](#), [log_pedantic](#), [MANAGEDBUF](#), [mm_alloc\(\)](#), [prime_field_write\(\)](#), [prime_user_signet_t](#), [PRIME_USER_SIGNING_REQUEST](#), [SECP256K1_KEY_PUB_LEN](#), [secp256k1_public_get\(\)](#), [secp256k1_public_set\(\)](#), [st_length_get\(\)](#), [st_write](#), and [user_signet_free\(\)](#).

Referenced by [prime_request_generate\(\)](#).

5.393.1.2 [stringer_t * user_request_get](#) ([prime_user_signet_t](#) * user, [stringer_t](#) * output)

Definition at line 221 of file [requests.c](#).

References [ED25519_KEY_PUB_LEN](#), [ed25519_public_get\(\)](#), [ED25519_SIGNATURE_LEN](#), [length](#), [log_pedantic](#), [MANAGEDBUF](#), [prime_field_write\(\)](#), [prime_header_user_signing_request_write\(\)](#), [PRIME_USER_SIGNET](#), [SECP256K1_KEY_PUB_LEN](#), [secp256k1_public_get\(\)](#), [st_alloc\(\)](#), [st_avail_get\(\)](#), [st_cleanup](#), [st_opt_get\(\)](#), [st_valid_destination\(\)](#), [st_wipe\(\)](#), [st_write](#), and [user_request_length\(\)](#).

Referenced by [prime_get\(\)](#).

5.393.1.3 [size_t user_request_length](#) ([prime_user_signet_t](#) * user)

Definition at line 10 of file [requests.c](#).

Referenced by [user_request_get\(\)](#).

5.393.1.4 [prime_user_signet_t * user_request_rotation](#) ([prime_user_key_t](#) * user, [prime_user_key_t](#) * previous)

Generate a user signet signing rotation request using the previous, and new user private key structures.

Definition at line 151 of file [requests.c](#).

References ED25519_KEY_PUB_LEN, ED25519_PRIV, ed25519_public_get(), ed25519_public_set(), ed25519_sign(), ED25519_SIGNATURE_LEN, log_pedantic, MANAGEDBUF, mm_alloc(), prime_field_write(), prime_user_signet_t, PRIME_USER_SIGNING_REQUEST, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), secp256k1_public_set(), st_length_get(), st_write, and user_signet_free().

Referenced by prime_request_generate().

5.393.1.5 prime_user_signet_t* user_request_set (stringer_t * user)

Definition at line 278 of file requests.c.

References ed25519_public_set(), log_pedantic, pl_data_get(), pl_length_get(), prime_field_get(), prime_field_t, prime_object_free(), prime_object_t, prime_unpack(), prime_user_signet_t, PRIME_USER_SIGNING_REQUEST, secp256k1_public_set(), st_import(), st_length_get(), user_request_verify_self(), user_signet_alloc(), and user_signet_free().

Referenced by prime_set().

5.393.1.6 prime_user_signet_t* user_request_sign (prime_user_signet_t * request, prime_org_key_t * org)

Accepts a user signet signing request and signs it using the provided org key, returning a valid signet.

Definition at line 28 of file requests.c.

References ED25519_KEY_PUB_LEN, ED25519_PRIV, ED25519_PUB, ed25519_public_get(), ed25519_public_set(), ed25519_sign(), ED25519_SIGNATURE_LEN, log_pedantic, MANAGEDBUF, mm_alloc(), prime_field_write(), PRIME_USER_SIGNET, prime_user_signet_t, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), secp256k1_public_set(), st_dupe(), st_empty, st_length_get(), st_write, user_request_verify_self(), and user_signet_free().

Referenced by prime_request_sign().

5.393.1.7 bool_t user_request_verify_chain_of_custody (prime_user_signet_t * user, prime_user_signet_t * previous)

Definition at line 379 of file requests.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate().

5.393.1.8 bool_t user_request_verify_self (prime_user_signet_t * user)

Definition at line 351 of file requests.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate(), user_request_set(), and user_request_sign().

5.394 src/providers/prime/signets/signets.h File Reference

Functions

- [prime_org_signet_t * org_signet_alloc](#) (void)

orgs.c

- [stringer_t * org_signet_fingerprint](#) (prime_org_signet_t *org, stringer_t *output)
- void [org_signet_free](#) (prime_org_signet_t *org)
- [prime_org_signet_t * org_signet_generate](#) (prime_org_key_t *org)

Derive an organizational signet from the corresponding private key structures.

- [stringer_t * org_signet_get](#) (prime_org_signet_t *org, stringer_t *output)
- size_t [org_signet_length](#) (prime_org_signet_t *org)
- [prime_org_signet_t * org_signet_set](#) (stringer_t *org)
- bool_t [org_signet_verify](#) (prime_org_signet_t *org)
- [prime_user_signet_t * user_signet_alloc](#) (void)

users.c

- [stringer_t * user_signet_fingerprint](#) (prime_user_signet_t *user, stringer_t *output)
- void [user_signet_free](#) (prime_user_signet_t *user)
- [stringer_t * user_signet_get](#) (prime_user_signet_t *user, stringer_t *output)
- size_t [user_signet_length](#) (prime_user_signet_t *user)
- [prime_user_signet_t * user_signet_set](#) (stringer_t *user)
- bool_t [user_signet_verify_chain_of_custody](#) (prime_user_signet_t *user, prime_user_signet_t *previous)
- bool_t [user_signet_verify_org](#) (prime_user_signet_t *user, prime_org_signet_t *org)
- bool_t [user_signet_verify_self](#) (prime_user_signet_t *user)
- [prime_user_signet_t * user_request_generate](#) (prime_user_key_t *user)

requests.c

- [stringer_t * user_request_get](#) (prime_user_signet_t *user, stringer_t *output)
- size_t [user_request_length](#) (prime_user_signet_t *user)
- [prime_user_signet_t * user_request_rotation](#) (prime_user_key_t *user, prime_user_key_t *previous)

Generate a user signet signing rotation request using the previous, and new user private key structures.

- [prime_user_signet_t * user_request_set](#) (stringer_t *user)
- [prime_user_signet_t * user_request_sign](#) (prime_user_signet_t *request, prime_org_key_t *org)

Accepts a user signet signing request and signs it using the provided org key, returning a valid signet.

- bool_t [user_request_verify_chain_of_custody](#) (prime_user_signet_t *user, prime_user_signet_t *previous)
- bool_t [user_request_verify_self](#) (prime_user_signet_t *user)

5.394.1 Function Documentation

5.394.1.1 prime_org_signet_t* org_signet_alloc (void)

orgs.c

Definition at line 24 of file orgs.c.

References [log_pedantic](#), [mm_alloc\(\)](#), [mm_wipe\(\)](#), and [prime_org_signet_t](#).

Referenced by [org_signet_set\(\)](#).

5.394.1.2 stringer_t* org_signet_fingerprint (prime_org_signet_t * org, stringer_t * output)

Definition at line 230 of file orgs.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, hash_sha512(), MANAGEDBUF, prime_field_write(), PRIME_ORG_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_fingerprint().

5.394.1.3 void org_signet_free (prime_org_signet_t * org)

Definition at line 12 of file orgs.c.

References ed25519_free(), mm_free(), secp256k1_free(), and st_free().

Referenced by org_signet_generate(), org_signet_set(), and prime_free().

5.394.1.4 prime_org_signet_t* org_signet_generate (prime_org_key_t * org)

Derive an organizational signet from the corresponding private key structures.

Definition at line 41 of file orgs.c.

References ED25519_KEY_PUB_LEN, ED25519_PRIV, ed25519_public_get(), ed25519_public_set(), ed25519_sign(), log_pedantic, MANAGEDBUF, mm_alloc(), org_signet_free(), prime_field_write(), PRIME_ORG_SIGNET, prime_org_signet_t, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), secp256k1_public_set(), st_cleanup, st_free(), and st_write.

Referenced by prime_signet_generate().

5.394.1.5 stringer_t* org_signet_get (prime_org_signet_t * org, stringer_t * output)

Definition at line 106 of file orgs.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, length, log_pedantic, MANAGEDBUF, org_signet_length(), prime_field_write(), prime_header_org_signet_write(), PRIME_ORG_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_alloc(), st_avail_get(), st_cleanup, st_opt_get(), st_valid_destination(), st_wipe(), and st_write.

Referenced by prime_get().

5.394.1.6 size_t org_signet_length (prime_org_signet_t * org)

Definition at line 97 of file orgs.c.

Referenced by org_signet_get().

5.394.1.7 prime_org_signet_t* org_signet_set (stringer_t * org)

Definition at line 150 of file orgs.c.

References ed25519_public_set(), log_pedantic, org_signet_alloc(), org_signet_free(), org_signet_verify(), pl_data_get(), pl_length_get(), prime_field_get(), prime_field_t, prime_object_free(), prime_object_t, PRIME_ORG_SIGNET, prime_org_signet_t, prime_unpack(), secp256k1_public_set(), st_import(), and st_length_get().

Referenced by prime_set().

5.394.1.8 bool_t org_signet_verify (prime_org_signet_t * org)

Definition at line 212 of file orgs.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ed25519_verify(), MANAGEDBUF, prime_field_write(), PRIME_ORG_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by org_signet_set(), and prime_signet_validate().

5.394.1.9 prime_user_signet_t* user_request_generate (prime_user_key_t * user)

[requests.c](#)

Definition at line 101 of file requests.c.

References ED25519_KEY_PUB_LEN, ED25519_PRIV, ed25519_public_get(), ed25519_public_set(), ed25519_sign(), log_pedantic, MANAGEDBUF, mm_alloc(), prime_field_write(), prime_user_signet_t, PRIME_USER_SIGNING_REQUEST, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), secp256k1_public_set(), st_length_get(), st_write, and user_signet_free().

Referenced by prime_request_generate().

5.394.1.10 stringer_t* user_request_get (prime_user_signet_t * user, stringer_t * output)

Definition at line 221 of file requests.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, length, log_pedantic, MANAGEDBUF, prime_field_write(), prime_header_user_signing_request_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_alloc(), st_avail_get(), st_cleanup, st_opt_get(), st_valid_destination(), st_wipe(), st_write, and user_request_length().

Referenced by prime_get().

5.394.1.11 size_t user_request_length (prime_user_signet_t * user)

Definition at line 10 of file requests.c.

Referenced by user_request_get().

5.394.1.12 prime_user_signet_t* user_request_rotation (prime_user_key_t * user, prime_user_key_t * previous)

Generate a user signet signing rotation request using the previous, and new user private key structures.

Definition at line 151 of file requests.c.

References ED25519_KEY_PUB_LEN, ED25519_PRIV, ed25519_public_get(), ed25519_public_set(), ed25519_sign(), ED25519_SIGNATURE_LEN, log_pedantic, MANAGEDBUF, mm_alloc(), prime_field_write(), prime_user_signet_t, PRIME_USER_SIGNING_REQUEST, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), secp256k1_public_set(), st_length_get(), st_write, and user_signet_free().

Referenced by prime_request_generate().

5.394.1.13 prime_user_signet_t* user_request_set (stringer_t * user)

Definition at line 278 of file requests.c.

References ed25519_public_set(), log_pedantic, pl_data_get(), pl_length_get(), prime_field_get(), prime_field_t, prime_object_free(), prime_object_t, prime_unpack(), prime_user_signet_t, PRIME_USER_SIGNING_REQUEST, secp256k1_public_set(), st_import(), st_length_get(), user_request_verify_self(), user_signet_alloc(), and user_signet_free().

Referenced by prime_set().

5.394.1.14 prime_user_signet_t* user_request_sign (prime_user_signet_t * request, prime_org_key_t * org)

Accepts a user signet signing request and signs it using the provided org key, returning a valid signet.

Definition at line 28 of file requests.c.

References ED25519_KEY_PUB_LEN, ED25519_PRIV, ED25519_PUB, ed25519_public_get(), ed25519_public_set(), ed25519_sign(), ED25519_SIGNATURE_LEN, log_pedantic, MANAGEDBUF, mm_alloc(), prime_field_write(), PRIME_USER_SIGNET, prime_user_signet_t, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), secp256k1_public_set(), st_dupe(), st_empty, st_length_get(), st_write, user_request_verify_self(), and user_signet_free().

Referenced by prime_request_sign().

5.394.1.15 **bool_t user_request_verify_chain_of_custody (prime_user_signet_t * user, prime_user_signet_t * previous)**

Definition at line 379 of file requests.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate().

5.394.1.16 **bool_t user_request_verify_self (prime_user_signet_t * user)**

Definition at line 351 of file requests.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate(), user_request_set(), and user_request_sign().

5.394.1.17 **prime_user_signet_t* user_signet_alloc (void)**

users.c

Definition at line 26 of file users.c.

References log_pedantic, mm_alloc(), mm_wipe(), and prime_user_signet_t.

Referenced by user_request_set(), and user_signet_set().

5.394.1.18 **stringer_t* user_signet_fingerprint (prime_user_signet_t * user, stringer_t * output)**

Definition at line 197 of file users.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, hash_sha512(), MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_fingerprint().

5.394.1.19 **void user_signet_free (prime_user_signet_t * user)**

Definition at line 12 of file users.c.

References ed25519_free(), mm_free(), secp256k1_free(), and st_free().

Referenced by prime_free(), user_request_generate(), user_request_rotation(), user_request_set(), user_request_sign(), and user_signet_set().

5.394.1.20 **stringer_t* user_signet_get (prime_user_signet_t * user, stringer_t * output)**

Definition at line 53 of file users.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, length, log_pedantic, MANAGEDBUF, prime_field_write(), prime_header_user_signet_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_alloc(), st_avail_get(), st_cleanup, st_opt_get(), st_valid_destination(), st_wipe(), st_write, and user_signet_length().

Referenced by prime_get().

5.394.1.21 size_t user_signet_length (prime_user_signet_t * user)

Definition at line 40 of file users.c.

Referenced by user_signet_get().

5.394.1.22 prime_user_signet_t* user_signet_set (stringer_t * user)

Definition at line 114 of file users.c.

References ed25519_public_set(), log_pedantic, pl_data_get(), pl_length_get(), prime_field_get(), prime_field_t, prime_object_free(), prime_object_t, prime_unpack(), PRIME_USER_SIGNET, prime_user_signet_t, secp256k1_public_set(), st_import(), st_length_get(), user_signet_alloc(), user_signet_free(), and user_signet_verify_self().

Referenced by prime_set().

5.394.1.23 bool_t user_signet_verify_chain_of_custody (prime_user_signet_t * user, prime_user_signet_t * previous)

Definition at line 221 of file users.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate().

5.394.1.24 bool_t user_signet_verify_org (prime_user_signet_t * user, prime_org_signet_t * org)

Definition at line 249 of file users.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate().

5.394.1.25 bool_t user_signet_verify_self (prime_user_signet_t * user)

Definition at line 279 of file users.c.

References ED25519_KEY_PUB_LEN, ed25519_public_get(), ED25519_SIGNATURE_LEN, ed25519_verify(), log_pedantic, MANAGEDBUF, prime_field_write(), PRIME_USER_SIGNET, SECP256K1_KEY_PUB_LEN, secp256k1_public_get(), st_length_get(), and st_write.

Referenced by prime_signet_validate(), and user_signet_set().

5.395 src/providers/prime/transposition/armored/armored.h File Reference

Functions

- [stringer_t * prime_pem_begin](#) (uint16_t type)
pem.c
- [stringer_t * prime_pem_end](#) (uint16_t type)
- [stringer_t * prime_pem_unwrap](#) (stringer_t *pem, stringer_t *output)
- [stringer_t * prime_pem_wrap](#) (stringer_t *object, stringer_t *output)

5.395.1 Function Documentation

5.395.1.1 stringer_t* prime_pem_begin (uint16_t type)

pem.c

Definition at line 36 of file pem.c.

References [log_pedantic](#), [PRIME_MESSAGE_ENCRYPTED](#), [PRIME_ORG_KEY](#), [PRIME_ORG_KEY_ENCRYPTED](#), [PRIME_ORG_SIGNET](#), [prime_pem_headings](#), [PRIME_USER_KEY](#), [PRIME_USER_KEY_ENCRYPTED](#), [PRIME_USER_SIGNET](#), and [PRIME_USER_SIGNING_REQUEST](#).

Referenced by [prime_pem_wrap\(\)](#).

5.395.1.2 stringer_t* prime_pem_end (uint16_t type)

Definition at line 73 of file pem.c.

References [log_pedantic](#), [PRIME_MESSAGE_ENCRYPTED](#), [PRIME_ORG_KEY](#), [PRIME_ORG_KEY_ENCRYPTED](#), [PRIME_ORG_SIGNET](#), [prime_pem_endings](#), [PRIME_USER_KEY](#), [PRIME_USER_KEY_ENCRYPTED](#), [PRIME_USER_SIGNET](#), and [PRIME_USER_SIGNING_REQUEST](#).

Referenced by [prime_pem_wrap\(\)](#).

5.395.1.3 stringer_t* prime_pem_unwrap (stringer_t * pem, stringer_t * output)

LOW: We check the length, but we don't confirm that the 4 chars following the = are valid base64 chars.

Definition at line 190 of file pem.c.

References [base64_decode\(\)](#), [crc24_checksum\(\)](#), [MANAGEDBUF](#), [pl_clone\(\)](#), [pl_data_get\(\)](#), [pl_empty\(\)](#), [pl_init\(\)](#), [pl_length_get\(\)](#), [pl_null\(\)](#), [pl_starts_with_char\(\)](#), [pl_trim\(\)](#), [placer_t](#), [prime_pem_endings](#), [prime_pem_headings](#), [st_cmp_cs_eq\(\)](#), [st_data_get\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [st_uchar_get\(\)](#), [tok_pop\(\)](#), [tok_pop_init_st\(\)](#), and [type\(\)](#).

Referenced by [prime_key_decrypt\(\)](#), and [prime_set\(\)](#).

5.395.1.4 stringer_t* prime_pem_wrap (stringer_t * object, stringer_t * output)

Definition at line 110 of file pem.c.

References [base64_encode_wrap\(\)](#), [base64_encoded_length_wrap\(\)](#), [crc24_checksum\(\)](#), [length](#), [log_pedantic](#), [MANAGEDBUF](#), [NULLER](#), [PLACER](#), [prime_header_read\(\)](#), [prime_object_type\(\)](#), [prime_pem_begin\(\)](#), [prime_pem_end\(\)](#), [PRIME_PEM_LINE_WRAP_CHARS](#), [PRIME_PEM_LINE_WRAP_LENGTH](#), [PRIME_PEM_LINE_WRAP_TYPE](#), [st_data_get\(\)](#), [st_dupes\(\)](#), [st_empty\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [st_output\(\)](#), [st_write\(\)](#), and [type\(\)](#).

Referenced by [prime_get\(\)](#), and [prime_key_encrypt\(\)](#).

5.396 src/providers/prime/transposition/armored/pem.c File Reference

```
#include "magma.h"
```

Defines

- `#define PRIME_PEM_LINE_WRAP_LENGTH 64`
- `#define PRIME_PEM_LINE_WRAP_CHARS "\n"`
- `#define PRIME_PEM_LINE_WRAP_TYPE BASE64_LINE_WRAP_LF`

Functions

- `stringer_t * prime_pem_begin (uint16_t type)`
pem.c
- `stringer_t * prime_pem_end (uint16_t type)`
- `stringer_t * prime_pem_wrap (stringer_t *object, stringer_t *output)`
- `stringer_t * prime_pem_unwrap (stringer_t *pem, stringer_t *output)`

Variables

- `stringer_t * prime_pem_headings []`
- `stringer_t * prime_pem_endings []`

5.396.1 Define Documentation

5.396.1.1 `#define PRIME_PEM_LINE_WRAP_CHARS "\n"`

Definition at line 11 of file pem.c.

Referenced by `prime_pem_wrap()`.

5.396.1.2 `#define PRIME_PEM_LINE_WRAP_LENGTH 64`

Definition at line 10 of file pem.c.

Referenced by `prime_pem_wrap()`.

5.396.1.3 `#define PRIME_PEM_LINE_WRAP_TYPE BASE64_LINE_WRAP_LF`

Definition at line 12 of file pem.c.

Referenced by `prime_pem_wrap()`.

5.396.2 Function Documentation

5.396.2.1 `stringer_t * prime_pem_begin (uint16_t type)`

pem.c

Definition at line 36 of file pem.c.

References `log_pedantic`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `prime_pem_headings`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_pem_wrap()`.

5.396.2.2 `stringer_t* prime_pem_end (uint16_t type)`

Definition at line 73 of file `pem.c`.

References `log_pedantic`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `prime_pem_endings`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_pem_wrap()`.

5.396.2.3 `stringer_t* prime_pem_unwrap (stringer_t * pem, stringer_t * output)`

LOW: We check the length, but we don't confirm that the 4 chars following the = are valid base64 chars.

Definition at line 190 of file `pem.c`.

References `base64_decode()`, `crc24_checksum()`, `MANAGEDBUF`, `pl_clone()`, `pl_data_get()`, `pl_empty()`, `pl_init()`, `pl_length_get()`, `pl_null()`, `pl_starts_with_char()`, `pl_trim()`, `placer_t`, `prime_pem_endings`, `prime_pem_headings`, `st_cmp_cs_eq()`, `st_data_get()`, `st_free()`, `st_length_get()`, `st_uchar_get()`, `tok_pop()`, `tok_pop_init_st()`, and `type()`.

Referenced by `prime_key_decrypt()`, and `prime_set()`.

5.396.2.4 `stringer_t* prime_pem_wrap (stringer_t * object, stringer_t * output)`

Definition at line 110 of file `pem.c`.

References `base64_encode_wrap()`, `base64_encoded_length_wrap()`, `crc24_checksum()`, `length`, `log_pedantic`, `MANAGEDBUF`, `NULLER`, `PLACER`, `prime_header_read()`, `prime_object_type()`, `prime_pem_begin()`, `prime_pem_end()`, `PRIME_PEM_LINE_WRAP_CHARS`, `PRIME_PEM_LINE_WRAP_LENGTH`, `PRIME_PEM_LINE_WRAP_TYPE`, `st_data_get()`, `st_dupe()`, `st_empty()`, `st_free()`, `st_length_get()`, `st_output()`, `st_write`, and `type()`.

Referenced by `prime_get()`, and `prime_key_encrypt()`.

5.396.3 Variable Documentation

5.396.3.1 `stringer_t* prime_pem_endings[]`

Initial value:

```
{
    CONSTANT("-----END USER KEY-----"),
    CONSTANT("-----END USER SIGNET-----"),
    CONSTANT("-----END SIGNET SIGNING REQUEST-----"),
    CONSTANT("-----END ORGANIZATIONAL KEY-----"),
    CONSTANT("-----END ORGANIZATIONAL SIGNET-----"),
    CONSTANT("-----END ENCRYPTED USER KEY-----"),
    CONSTANT("-----END ENCRYPTED ORGANIZATIONAL KEY-----"),
}
```

Definition at line 25 of file `pem.c`.

Referenced by `prime_pem_end()`, and `prime_pem_unwrap()`.

5.396.3.2 stringer_t* prime_pem_headings[]

Initial value:

```
{  
    CONSTANT("-----BEGIN USER KEY-----"),  
    CONSTANT("-----BEGIN USER SIGNET-----"),  
    CONSTANT("-----BEGIN SIGNET SIGNING REQUEST-----"),  
    CONSTANT("-----BEGIN ORGANIZATIONAL KEY-----"),  
    CONSTANT("-----BEGIN ORGANIZATIONAL SIGNET-----"),  
    CONSTANT("-----BEGIN ENCRYPTED USER KEY-----"),  
    CONSTANT("-----BEGIN ENCRYPTED ORGANIZATIONAL KEY-----"),  
}
```

Definition at line 14 of file pem.c.

Referenced by prime_pem_begin(), and prime_pem_unwrap().

5.397 src/providers/prime/transposition/binary/binary.h File Reference

Typedefs

- typedef uint32_t [prime_size_t](#)
Object data types.
- typedef uint8_t [prime_field_type_t](#)
Field data types.
- typedef [placer_t](#) [prime_field_data_t](#)

Functions

- struct [__attribute__](#) ((packed))
- [prime_object_t](#) * [prime_unpack](#) ([stringer_t](#) *data)
unpack.c
- [int_t](#) [prime_unpack_fields](#) ([prime_object_t](#) *object, [stringer_t](#) *fields)
- [prime_size_t](#) [prime_unpack_validate](#) ([stringer_t](#) *fields)
- [int_t](#) [prime_reader_open](#) ([stringer_t](#) *data, [prime_reader_t](#) *reader)
reader.c
- [int_t](#) [prime_reader_payload](#) ([prime_reader_t](#) *reader, [int_t](#) bytes, [placer_t](#) *payload)
Read in the field payload and return the result wrapped up by a place holder.
- [int_t](#) [prime_reader_size](#) ([prime_reader_t](#) *reader, [int_t](#) bytes)
Read 1, 2 or 3 bytes representing a the payload size for a field, and convert the bytes from big endian into the native format before returning the result.
- [int_t](#) [prime_reader_type](#) ([prime_reader_t](#) *reader)
Read a single byte using the reader and convert it to a type field.
- [prime_object_t](#) * [prime_object_alloc](#) ([uint16_t](#) type, [prime_size_t](#) size, [prime_size_t](#) fields)
objects.c
- void [prime_object_free](#) ([prime_object_t](#) *object)
- [size_t](#) [prime_object_size_max](#) ([uint16_t](#) type)
- [size_t](#) [prime_object_size_min](#) ([uint16_t](#) type)
- [chr_t](#) * [prime_object_type](#) ([uint16_t](#) type)
- [stringer_t](#) * [prime_header_encrypted_message_write](#) ([size_t](#) size, [stringer_t](#) *output)
headers.c
- [stringer_t](#) * [prime_header_encrypted_org_key_write](#) ([size_t](#) size, [stringer_t](#) *output)
- [stringer_t](#) * [prime_header_encrypted_user_key_write](#) ([size_t](#) size, [stringer_t](#) *output)
- [size_t](#) [prime_header_length](#) ([uint16_t](#) type)
Determine whether a given type uses a 5 or 6 byte header.
- [stringer_t](#) * [prime_header_org_key_write](#) ([size_t](#) size, [stringer_t](#) *output)
- [stringer_t](#) * [prime_header_org_signet_write](#) ([size_t](#) size, [stringer_t](#) *output)
- [int_t](#) [prime_header_read](#) ([stringer_t](#) *object, [uint16_t](#) *type, [uint32_t](#) *size)
Read the first few bytes of a serialized PRIME object and return the magic number and object size in native form.

- [stringer_t * prime_header_user_key_write](#) (size_t size, [stringer_t](#) *output)
 - [stringer_t * prime_header_user_signet_write](#) (size_t size, [stringer_t](#) *output)
 - [stringer_t * prime_header_user_signing_request_write](#) (size_t size, [stringer_t](#) *output)
 - [stringer_t * prime_header_write](#) (uint16_t type, size_t size, [stringer_t](#) *output)
 - [prime_field_t * prime_field_get](#) ([prime_object_t](#) *object, [prime_field_type_t](#) type)
- fields.c*
- [int_t prime_field_size_length](#) ([prime_field_type_t](#) field)
 - [size_t prime_field_size_max](#) (uint16_t type, [prime_field_type_t](#) field)
 - [stringer_t * prime_field_write](#) (uint16_t type, [prime_field_type_t](#) field, size_t size, [stringer_t](#) *data, [stringer_t](#) *output)

Variables

- [prime_field_t](#)
- [prime_object_t](#)
- [prime_reader_t](#)

5.397.1 Typedef Documentation

5.397.1.1 typedef placer_t prime_field_data_t

Definition at line 16 of file binary.h.

5.397.1.2 typedef uint8_t prime_field_type_t

Field data types.

Definition at line 15 of file binary.h.

5.397.1.3 typedef uint32_t prime_size_t

Object data types.

Definition at line 12 of file binary.h.

5.397.2 Function Documentation

5.397.2.1 struct __attribute__((packed)) [read]

- < The field type identifier. (1b)
- < The field payload represented by a placeholder.
- < The object header.
- < The object type identifier. (2b)
- < The size of the object data (not including the header). (3b or 4b)
- < The array of parsed fields inside the packed object buffer.

Definition at line 46 of file binary.h.

References [placer_t](#).

5.397.2.2 `prime_field_t* prime_field_get (prime_object_t * object, prime_field_type_t type)`

[fields.c](#)

Definition at line 10 of file `fields.c`.

References `prime_field_t`.

Referenced by `org_key_set()`, `org_signet_set()`, `user_key_set()`, `user_request_set()`, and `user_signet_set()`.

5.397.2.3 `int_t prime_field_size_length (prime_field_type_t field)`

TODO: else if (`field == 251`) `result = ???`;

Definition at line 30 of file `fields.c`.

Referenced by `prime_field_write()`, `prime_unpack_fields()`, and `prime_unpack_validate()`.

5.397.2.4 `size_t prime_field_size_max (uint16_t type, prime_field_type_t field)`

TODO: else if (`field == 251`) `result = ???`;

TODO: else if (`field == 251`) `result = ???`;

Definition at line 54 of file `fields.c`.

References `log_pedantic`, `PRIME_FIXED_SIZE`, `PRIME_MAX_1_BYTE`, `PRIME_MAX_2_BYTE`, `PRIME_MAX_3_BYTE`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_field_write()`.

5.397.2.5 `stringer_t* prime_field_write (uint16_t type, prime_field_type_t field, size_t size, stringer_t * data, stringer_t * output)`

TODO: Add undefined field support.

Definition at line 102 of file `fields.c`.

References `log_error`, `log_pedantic`, `mm_copy()`, `prime_field_size_length()`, `prime_field_size_max()`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, `st_data_get()`, `st_empty_out()`, `st_free()`, `st_length_set()`, `st_opt_get()`, `st_output()`, and `st_valid_tracked()`.

Referenced by `ephemeral_chunk_get()`, `org_key_get()`, `org_signet_fingerprint()`, `org_signet_generate()`, `org_signet_get()`, `org_signet_verify()`, `user_key_get()`, `user_request_generate()`, `user_request_get()`, `user_request_rotation()`, `user_request_sign()`, `user_request_verify_chain_of_custody()`, `user_request_verify_self()`, `user_signet_fingerprint()`, `user_signet_get()`, `user_signet_verify_chain_of_custody()`, `user_signet_verify_org()`, and `user_signet_verify_self()`.

5.397.2.6 `stringer_t* prime_header_encrypted_message_write (size_t size, stringer_t * output)`

[headers.c](#)

Definition at line 221 of file `headers.c`.

References `prime_header_write()`, and `PRIME_MESSAGE_ENCRYPTED`.

5.397.2.7 `stringer_t* prime_header_encrypted_org_key_write (size_t size, stringer_t * output)`

Definition at line 213 of file `headers.c`.

References `prime_header_write()`, and `PRIME_ORG_KEY_ENCRYPTED`.

5.397.2.8 stringer_t* prime_header_encrypted_user_key_write (size_t size, stringer_t * output)

Definition at line 217 of file headers.c.

References prime_header_write(), and PRIME_USER_KEY_ENCRYPTED.

5.397.2.9 size_t prime_header_length (uint16_t type)

Determine whether a given type uses a 5 or 6 byte header.

Parameters:

type The PRIME type.

Returns:

Returns the number of bytes used by the header, or 0 if an invalid type is supplied.

Definition at line 15 of file headers.c.

References length, log_pedantic, PRIME_MESSAGE_ENCRYPTED, PRIME_ORG_KEY, PRIME_ORG_KEY_ENCRYPTED, PRIME_ORG_SIGNET, PRIME_USER_KEY, PRIME_USER_KEY_ENCRYPTED, PRIME_USER_SIGNET, and PRIME_USER_SIGNING_REQUEST.

Referenced by prime_header_write().

5.397.2.10 stringer_t* prime_header_org_key_write (size_t size, stringer_t * output)

Definition at line 205 of file headers.c.

References prime_header_write(), and PRIME_ORG_KEY.

Referenced by org_key_get().

5.397.2.11 stringer_t* prime_header_org_signet_write (size_t size, stringer_t * output)

Definition at line 193 of file headers.c.

References prime_header_write(), and PRIME_ORG_SIGNET.

Referenced by org_signet_get().

5.397.2.12 int_t prime_header_read (stringer_t * object, uint16_t * type, uint32_t * size)

Read the first few bytes of a serialized PRIME object and return the magic number and object size in native form.

Parameters:

data The serialized object buffer.

type A pointer where the type will be stored, if successful.

size A pointer to where the parsed size will be stored, if the read is successful.

Returns:

0 if successful, negative values if the object data is invalid, or positive numbers if a programmatic error occurs. 2 = The object buffer was empty. 1 = A null pointer was supplied. 0 = Success. -1 = The buffer is too short for a valid PRIME object header. -2 = The size in the header doesn't match the amount of data supplied. (size = header_len + object_len). -3 = The header indicates an unrecognized PRIME type.

Definition at line 53 of file headers.c.

References `log_pedantic`, `mm_copy()`, `PRIME_MESSAGE_ABUSE`, `PRIME_MESSAGE_BOUNCE`, `PRIME_MESSAGE_DRAFT`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_MESSAGE_FORWARD`, `PRIME_MESSAGE_NAKED`, `PRIME_MESSAGE_SENT`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, and `st_empty_out()`.

Referenced by `aes_artifact_encrypt()`, `naked_message_get()`, `prime_key_decrypt()`, `prime_pem_wrap()`, `prime_set()`, and `prime_unpack()`.

5.397.2.13 `stringer_t* prime_header_user_key_write (size_t size, stringer_t * output)`

Definition at line 209 of file headers.c.

References `prime_header_write()`, and `PRIME_USER_KEY`.

Referenced by `user_key_get()`.

5.397.2.14 `stringer_t* prime_header_user_signet_write (size_t size, stringer_t * output)`

Definition at line 197 of file headers.c.

References `prime_header_write()`, and `PRIME_USER_SIGNET`.

Referenced by `user_signet_get()`.

5.397.2.15 `stringer_t* prime_header_user_signing_request_write (size_t size, stringer_t * output)`

Definition at line 201 of file headers.c.

References `prime_header_write()`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `user_request_get()`.

5.397.2.16 `stringer_t* prime_header_write (uint16_t type, size_t size, stringer_t * output)`

Parameters:

type
size
output

Returns:

Definition at line 127 of file headers.c.

References `length`, `log_error`, `log_pedantic`, `mm_copy()`, `prime_header_length()`, `PRIME_MESSAGE_ENCRYPTED`, `prime_object_size_max()`, `prime_object_size_min()`, `prime_object_type()`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, `PRIME_USER_SIGNING_REQUEST`, `st_data_get()`, `st_free()`, `st_length_set()`, `st_opt_get()`, `st_output()`, and `st_valid_tracked()`.

Referenced by `aes_artifact_decrypt()`, `prime_header_encrypted_message_write()`, `prime_header_encrypted_org_key_write()`, `prime_header_encrypted_user_key_write()`, `prime_header_org_key_write()`, `prime_header_org_signet_write()`, `prime_header_user_key_write()`, `prime_header_user_signet_write()`, and `prime_header_user_signing_request_write()`.

5.397.2.17 `prime_object_t* prime_object_alloc (uint16_t type, prime_size_t size, prime_size_t fields)`

objects.c

Definition at line 66 of file objects.c.

References `log_pedantic`, `mm_alloc()`, `mm_wipe()`, `prime_field_t`, and `prime_object_t`.

Referenced by `prime_unpack()`.

5.397.2.18 `void prime_object_free (prime_object_t * object)`

Definition at line 57 of file objects.c.

References `mm_free()`.

Referenced by `org_key_set()`, `org_signet_set()`, `prime_unpack()`, `user_key_set()`, `user_request_set()`, and `user_signet_set()`.

5.397.2.19 `size_t prime_object_size_max (uint16_t type)`

Definition at line 87 of file objects.c.

References `log_pedantic`, `PRIME_MAX_3_BYTE`, `PRIME_MAX_4_BYTE`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_header_write()`.

5.397.2.20 `size_t prime_object_size_min (uint16_t type)`

Definition at line 111 of file objects.c.

References `log_pedantic`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_header_write()`.

5.397.2.21 `chr_t* prime_object_type (uint16_t type)`

Definition at line 21 of file objects.c.

References `log_pedantic`, `PRIME_MESSAGE_ENCRYPTED`, `PRIME_ORG_KEY`, `PRIME_ORG_KEY_ENCRYPTED`, `PRIME_ORG_SIGNET`, `prime_types`, `PRIME_USER_KEY`, `PRIME_USER_KEY_ENCRYPTED`, `PRIME_USER_SIGNET`, and `PRIME_USER_SIGNING_REQUEST`.

Referenced by `prime_header_write()`, and `prime_pem_wrap()`.

5.397.2.22 `int_t prime_reader_open (stringer_t * data, prime_reader_t * reader)`

[reader.c](#)

Definition at line 81 of file reader.c.

References `pl_init()`, `st_data_get()`, `st_empty`, and `st_length_get()`.

Referenced by `prime_unpack_fields()`, and `prime_unpack_validate()`.

5.397.2.23 `int_t prime_reader_payload (prime_reader_t * reader, int_t bytes, placer_t * payload)`

Read in the field payload and return the result wrapped up by a place holder.

Parameters:

reader

Returns:

-1 if an error occurs, or a 0 if the payload is packaged and returned successfully.

Definition at line 61 of file reader.c.

References `pl_init()`, and `pl_null()`.

Referenced by `prime_unpack_fields()`, and `prime_unpack_validate()`.

5.397.2.24 `int_t prime_reader_size (prime_reader_t * reader, int_t bytes)`

Read 1, 2 or 3 bytes representing a the payload size for a field, and convert the bytes from big endian into the native format before returning the result.

Parameters:

reader

Returns:

-1 if an error occurs, or a field type between 1 and 255 if the reader contained data.

Definition at line 39 of file reader.c.

References `mm_copy()`.

Referenced by `prime_unpack_fields()`, and `prime_unpack_validate()`.

5.397.2.25 `int_t prime_reader_type (prime_reader_t * reader)`

Read a single byte using the reader and convert it to a type field.

Parameters:

reader

Returns:

-1 if an error occurs, 0 if the end of the buffer is reached, or a field type between 1 and 255 if the reader contained data.

Definition at line 16 of file reader.c.

Referenced by `prime_unpack_fields()`, and `prime_unpack_validate()`.

5.397.2.26 `prime_object_t* prime_unpack (stringer_t * data)`

[unpack.c](#)

Definition at line 124 of file unpack.c.

References `count`, `PLACER`, `prime_header_read()`, `PRIME_MESSAGE_ENCRYPTED`, `prime_object_alloc()`, `prime_object_free()`, `prime_object_t`, `PRIME_ORG_KEY_ENCRYPTED`, `prime_unpack_fields()`, `prime_unpack_validate()`, `PRIME_USER_KEY_ENCRYPTED`, `st_data_get()`, `st_length_get()`, and `type()`.

Referenced by `org_key_set()`, `org_signet_set()`, `user_key_set()`, `user_request_set()`, and `user_signet_set()`.

5.397.2.27 `int_t prime_unpack_fields (prime_object_t * object, stringer_t * fields)`

Definition at line 65 of file unpack.c.

References bytes, count, ED25519_SIGNATURE_LEN, placer_t, prime_field_size_length(), prime_reader_open(), prime_reader_payload(), prime_reader_size(), prime_reader_t, prime_reader_type(), and type().

Referenced by prime_unpack().

5.397.2.28 prime_size_t prime_unpack_validate (stringer_t * *fields*)

Definition at line 10 of file unpack.c.

References bytes, count, ED25519_SIGNATURE_LEN, placer_t, prime_field_size_length(), prime_reader_open(), prime_reader_payload(), prime_reader_size(), prime_reader_t, prime_reader_type(), and type().

Referenced by prime_unpack().

5.397.3 Variable Documentation

5.397.3.1 prime_field_t

Definition at line 26 of file binary.h.

Referenced by org_key_set(), org_signet_set(), prime_field_get(), prime_object_alloc(), user_key_set(), user_request_set(), and user_signet_set().

5.397.3.2 prime_object_t

Definition at line 41 of file binary.h.

Referenced by org_key_set(), org_signet_set(), prime_object_alloc(), prime_unpack(), user_key_set(), user_request_set(), and user_signet_set().

5.397.3.3 prime_reader_t

Definition at line 50 of file binary.h.

Referenced by prime_unpack_fields(), and prime_unpack_validate().

5.398 src/providers/prime/transposition/binary/fields.c File Reference

```
#include "magma.h"
```

Functions

- [prime_field_t * prime_field_get](#) ([prime_object_t](#) *object, [prime_field_type_t](#) type)
fields.c
- [int_t prime_field_size_length](#) ([prime_field_type_t](#) field)
- [size_t prime_field_size_max](#) ([uint16_t](#) type, [prime_field_type_t](#) field)
- [stringer_t * prime_field_write](#) ([uint16_t](#) type, [prime_field_type_t](#) field, [size_t](#) size, [stringer_t](#) *data, [stringer_t](#) *output)

5.398.1 Function Documentation

5.398.1.1 [prime_field_t * prime_field_get](#) ([prime_object_t](#) * *object*, [prime_field_type_t](#) *type*)

[fields.c](#)

Definition at line 10 of file [fields.c](#).

References [prime_field_t](#).

Referenced by [org_key_set\(\)](#), [org_signet_set\(\)](#), [user_key_set\(\)](#), [user_request_set\(\)](#), and [user_signet_set\(\)](#).

5.398.1.2 [int_t prime_field_size_length](#) ([prime_field_type_t](#) *field*)

TODO: else if (field == 251) result = ???;

Definition at line 30 of file [fields.c](#).

Referenced by [prime_field_write\(\)](#), [prime_unpack_fields\(\)](#), and [prime_unpack_validate\(\)](#).

5.398.1.3 [size_t prime_field_size_max](#) ([uint16_t](#) *type*, [prime_field_type_t](#) *field*)

TODO: else if (field == 251) result = ???;

TODO: else if (field == 251) result = ???;

Definition at line 54 of file [fields.c](#).

References [log_pedantic](#), [PRIME_FIXED_SIZE](#), [PRIME_MAX_1_BYTE](#), [PRIME_MAX_2_BYTE](#), [PRIME_MAX_3_BYTE](#), [PRIME_MESSAGE_ENCRYPTED](#), [PRIME_ORG_KEY](#), [PRIME_ORG_KEY_ENCRYPTED](#), [PRIME_ORG_SIGNET](#), [PRIME_USER_KEY](#), [PRIME_USER_KEY_ENCRYPTED](#), [PRIME_USER_SIGNET](#), and [PRIME_USER_SIGNING_REQUEST](#).

Referenced by [prime_field_write\(\)](#).

5.398.1.4 [stringer_t * prime_field_write](#) ([uint16_t](#) *type*, [prime_field_type_t](#) *field*, [size_t](#) *size*, [stringer_t](#) * *data*, [stringer_t](#) * *output*)

TODO: Add undefined field support.

Definition at line 102 of file [fields.c](#).

References [log_error](#), [log_pedantic](#), [mm_copy\(\)](#), [prime_field_size_length\(\)](#), [prime_field_size_max\(\)](#), [PRIME_MESSAGE_ENCRYPTED](#), [PRIME_ORG_KEY](#), [PRIME_ORG_KEY_ENCRYPTED](#), [PRIME_ORG_SIGNET](#), [PRIME_USER_KEY](#), [PRIME_USER_KEY_ENCRYPTED](#), [PRIME_USER_SIGNET](#), [PRIME_USER_SIGNING_REQUEST](#), [st_data_get\(\)](#), [st_empty_out\(\)](#), [st_free\(\)](#), [st_length_set\(\)](#), [st_opt_get\(\)](#), [st_output\(\)](#), and [st_valid_tracked\(\)](#).

Referenced by `ephemeral_chunk_get()`, `org_key_get()`, `org_signet_fingerprint()`, `org_signet_generate()`, `org_signet_get()`, `org_signet_verify()`, `user_key_get()`, `user_request_generate()`, `user_request_get()`, `user_request_rotation()`, `user_request_sign()`, `user_request_verify_chain_of_custody()`, `user_request_verify_self()`, `user_signet_fingerprint()`, `user_signet_get()`, `user_signet_verify_chain_of_custody()`, `user_signet_verify_org()`, and `user_signet_verify_self()`.

5.399 src/providers/prime/transposition/binary/reader.c File Reference

```
#include "magma.h"
```

Functions

- [int_t prime_reader_type](#) ([prime_reader_t](#) *reader)
Read a single byte using the reader and convert it to a type field.
- [int_t prime_reader_size](#) ([prime_reader_t](#) *reader, [int_t](#) bytes)
Read 1, 2 or 3 bytes representing a the payload size for a field, and convert the bytes from big endian into the native format before returning the result.
- [int_t prime_reader_payload](#) ([prime_reader_t](#) *reader, [int_t](#) bytes, [placer_t](#) *payload)
Read in the field payload and return the result wrapped up by a place holder.
- [int_t prime_reader_open](#) ([stringer_t](#) *data, [prime_reader_t](#) *reader)
reader.c

5.399.1 Function Documentation

5.399.1.1 [int_t prime_reader_open](#) ([stringer_t](#) * data, [prime_reader_t](#) * reader)

[reader.c](#)

Definition at line 81 of file [reader.c](#).

References [pl_init\(\)](#), [st_data_get\(\)](#), [st_empty](#), and [st_length_get\(\)](#).

Referenced by [prime_unpack_fields\(\)](#), and [prime_unpack_validate\(\)](#).

5.399.1.2 [int_t prime_reader_payload](#) ([prime_reader_t](#) * reader, [int_t](#) bytes, [placer_t](#) * payload)

Read in the field payload and return the result wrapped up by a place holder.

Parameters:

reader

Returns:

-1 if an error occurs, or a 0 if the payload is packaged and returned sucessfully.

Definition at line 61 of file [reader.c](#).

References [pl_init\(\)](#), and [pl_null\(\)](#).

Referenced by [prime_unpack_fields\(\)](#), and [prime_unpack_validate\(\)](#).

5.399.1.3 [int_t prime_reader_size](#) ([prime_reader_t](#) * reader, [int_t](#) bytes)

Read 1, 2 or 3 bytes representing a the payload size for a field, and convert the bytes from big endian into the native format before returning the result.

Parameters:

reader

Returns:

-1 if an error occurs, or a field type between 1 and 255 if the reader contained data.

Definition at line 39 of file reader.c.

References `mm_copy()`.

Referenced by `prime_unpack_fields()`, and `prime_unpack_validate()`.

5.399.1.4 int_t prime_reader_type (prime_reader_t * reader)

Read a single byte using the reader and convert it to a type field.

Parameters:

reader

Returns:

-1 if an error occurs, 0 if the end of the buffer is reached, or a field type between 1 and 255 if the reader contained data.

Definition at line 16 of file reader.c.

Referenced by `prime_unpack_fields()`, and `prime_unpack_validate()`.

5.400 src/providers/prime/transposition/binary/unpack.c File Reference

```
#include "magma.h"
```

Functions

- [prime_size_t prime_unpack_validate \(stringer_t *fields\)](#)
- [int_t prime_unpack_fields \(prime_object_t *object, stringer_t *fields\)](#)
- [prime_object_t * prime_unpack \(stringer_t *data\)](#)

[unpack.c](#)

5.400.1 Function Documentation

5.400.1.1 prime_object_t* prime_unpack (stringer_t * data)

[unpack.c](#)

Definition at line 124 of file unpack.c.

References `count`, `PLACER`, `prime_header_read()`, `PRIME_MESSAGE_ENCRYPTED`, `prime_object_alloc()`, `prime_object_free()`, `prime_object_t`, `PRIME_ORG_KEY_ENCRYPTED`, `prime_unpack_fields()`, `prime_unpack_validate()`, `PRIME_USER_KEY_ENCRYPTED`, `st_data_get()`, `st_length_get()`, and `type()`.

Referenced by `org_key_set()`, `org_signet_set()`, `user_key_set()`, `user_request_set()`, and `user_signet_set()`.

5.400.1.2 int_t prime_unpack_fields (prime_object_t * object, stringer_t * fields)

Definition at line 65 of file unpack.c.

References `bytes`, `count`, `ED25519_SIGNATURE_LEN`, `placer_t`, `prime_field_size_length()`, `prime_reader_open()`, `prime_reader_payload()`, `prime_reader_size()`, `prime_reader_t`, `prime_reader_type()`, and `type()`.

Referenced by `prime_unpack()`.

5.400.1.3 prime_size_t prime_unpack_validate (stringer_t * fields)

Definition at line 10 of file unpack.c.

References `bytes`, `count`, `ED25519_SIGNATURE_LEN`, `placer_t`, `prime_field_size_length()`, `prime_reader_open()`, `prime_reader_payload()`, `prime_reader_size()`, `prime_reader_t`, `prime_reader_type()`, and `type()`.

Referenced by `prime_unpack()`.

5.401 src/providers/prime/transposition/transposition.h File Reference

```
#include "binary/binary.h"
#include "armored/armored.h"
```

Defines

- #define [PRIME_FIXED_SIZE](#) 64
- #define [PRIME_MAX_1_BYTE](#) 255
- #define [PRIME_MAX_2_BYTE](#) 65535
- #define [PRIME_MAX_3_BYTE](#) 16777215
- #define [PRIME_MAX_4_BYTE](#) 4294967295

5.401.1 Define Documentation

5.401.1.1 #define [PRIME_FIXED_SIZE](#) 64

Definition at line 11 of file transposition.h.

Referenced by [prime_field_size_max\(\)](#).

5.401.1.2 #define [PRIME_MAX_1_BYTE](#) 255

Definition at line 12 of file transposition.h.

Referenced by [prime_field_size_max\(\)](#).

5.401.1.3 #define [PRIME_MAX_2_BYTE](#) 65535

Definition at line 13 of file transposition.h.

Referenced by [prime_field_size_max\(\)](#).

5.401.1.4 #define [PRIME_MAX_3_BYTE](#) 16777215

Definition at line 14 of file transposition.h.

Referenced by [chunk_header_write\(\)](#), [prime_field_size_max\(\)](#), and [prime_object_size_max\(\)](#).

5.401.1.5 #define [PRIME_MAX_4_BYTE](#) 4294967295

Definition at line 15 of file transposition.h.

Referenced by [prime_object_size_max\(\)](#).

5.402 src/providers/providers.h File Reference

```
#include "database/database.h"
#include "consumers/consumers.h"
#include "checkers/checkers.h"
#include "compress/compress.h"
#include "cryptography/cryptography.h"
#include "stacie/stacie.h"
#include "prime/prime.h"
#include "parsers/parsers.h"
#include "storage/storage.h"
#include "images/images.h"
```

Functions

- `struct __attribute__((packed))`
- `bool_t lib_load (void)`
Load magmad.so dynamically and resolve all external dependencies from 3rd party providers.
- `void lib_unload (void)`
Unload magmad.so from memory.
- `bool_t lib_symbols (size_t count, symbol_t symbols[])`

Variables

- `symbol_t`

5.402.1 Function Documentation

5.402.1.1 `struct __attribute__((packed))` [read]

Definition at line 11 of file providers.h.

5.402.1.2 `bool_t lib_load (void)`

Load magmad.so dynamically and resolve all external dependencies from 3rd party providers.

Returns:

false on failure or true on success.

Definition at line 658 of file symbols.c.

References `build_commit()`, `build_stamp()`, `build_version()`, `magma_t::config`, `CONSTANT`, `magma_t::file`, `host_platform()`, `host_version()`, `lib_load_bzip()`, `lib_load_cache()`, `lib_load_clamav()`, `lib_load_dkim()`, `lib_load_dspam()`, `lib_load_freetype()`, `lib_load_gd()`, `lib_load_jansson()`, `lib_load_jpeg()`, `lib_load_lzo()`, `lib_load_mysql()`, `lib_load_openssl()`, `lib_load_png()`, `lib_load_spf()`, `lib_load_tokyo()`, `lib_load_utf8proc()`, `lib_load_xml()`, `lib_load_zlib()`, `lib_magma`, `lib_version_bzip()`, `lib_version_cache()`, `lib_version_clamav()`, `lib_version_dkim()`, `lib_version_dspam()`, `lib_version_freetype()`, `lib_version_gd()`, `lib_version_jansson()`, `lib_version_jpeg()`, `lib_version_lzo()`, `lib_version_mysql()`, `lib_version_openssl()`, `lib_version_png()`, `lib_version_spf()`, `lib_version_tokyo()`, `lib_version_utf8proc()`, `lib_version_`

xml(), lib_version_zlib(), magma_t::library, log_critical, log_options, M_LOG_FILE_DISABLE, M_LOG_FUNCTION_DISABLE, M_LOG_INFO, M_LOG_LINE_DISABLE, M_LOG_STACK_TRACE_DISABLE, M_LOG_TIME_DISABLE, magma, MANAGEDBUF, ns_empty(), NULLER, serv_charset_mysql(), serv_schema_mysql(), serv_type_mysql(), serv_version_mysql(), st_char_get(), and st_cmp_ci_eq().

Referenced by process_start().

5.402.1.3 bool_t lib_symbols (size_t count, symbol_t symbols[])

Referenced by lib_load_bzip(), lib_load_cache(), lib_load_clamav(), lib_load_dkim(), lib_load_dspam(), lib_load_freetype(), lib_load_gd(), lib_load_jansson(), lib_load_jpeg(), lib_load_lzo(), lib_load_mysql(), lib_load_openssl(), lib_load_png(), lib_load_spf(), lib_load_tokyo(), lib_load_utf8proc(), lib_load_xml(), and lib_load_zlib().

5.402.1.4 void lib_unload (void)

Unload magmad.so from memory.

Returns:

This function returns no value.

Definition at line 645 of file symbols.c.

References lib_magma, magma_t::library, magma, and magma_t::unload.

Referenced by process_stop().

5.402.2 Variable Documentation

5.402.2.1 symbol_t

Definition at line 14 of file providers.h.

Referenced by lib_load_bzip(), lib_load_cache(), lib_load_clamav(), lib_load_dkim(), lib_load_dspam(), lib_load_freetype(), lib_load_gd(), lib_load_jansson(), lib_load_jpeg(), lib_load_lzo(), lib_load_mysql(), lib_load_openssl(), lib_load_png(), lib_load_spf(), lib_load_tokyo(), lib_load_utf8proc(), lib_load_xml(), and lib_load_zlib().

5.403 src/providers/stacie/creation.c File Reference

Create cryptographically strong random STACIE salt, nonce, and shard values. `#include "magma.h"`

Functions

- `stringer_t * stacie_create_shard (stringer_t *output)`
Generates a random shard of precisely 64 bytes, suitable for use with a STACIE realm.
- `stringer_t * stacie_create_salt (stringer_t *output)`
Generates a random salt of precisely 128 bytes, suitable for use with STACIE authentication scheme.
- `stringer_t * stacie_create_nonce (stringer_t *output)`
Generates a random nonce of precisely 128 bytes, suitable for use with the STACIE authentication scheme.

5.403.1 Detailed Description

Create cryptographically strong random STACIE salt, nonce, and shard values.

Definition in file [creation.c](#).

5.403.2 Function Documentation

5.403.2.1 `stringer_t* stacie_create_nonce (stringer_t * output)`

Generates a random nonce of precisely 128 bytes, suitable for use with the STACIE authentication scheme. [creation.c](#)

Note:

While the salt and nonce creation functions are nearly identical, they remain separated so they can use distinct preprocessor definitions for the result length.

Parameters:

output a managed string to receive the output; if passed as NULL, an output buffer will be allocated.

Returns:

A managed string holding the freshly generated nonce in binary form. If NULL was passed in then the result must be freed by the caller.

Definition at line 91 of file [creation.c](#).

References [log_pedantic](#), [PLACER](#), [rand_write\(\)](#), [st_alloc\(\)](#), [st_avail_get\(\)](#), [st_cleanup](#), [st_data_get\(\)](#), [st_length_set\(\)](#), [st_opt_get\(\)](#), [st_valid_destination\(\)](#), [st_valid_tracked\(\)](#), and [STACIE_NONCE_LENGTH](#).

Referenced by [auth_challenge\(\)](#), and [auth_response\(\)](#).

5.403.2.2 `stringer_t* stacie_create_salt (stringer_t * output)`

Generates a random salt of precisely 128 bytes, suitable for use with STACIE authentication scheme.

Note:

While the salt and nonce creation functions are nearly identical, they remain separated so they can use distinct preprocessor definitions for the result length.

Parameters:

output a managed string to receive the output; if passed as NULL, an output buffer will be allocated.

Returns:

A managed string holding the freshly generated salt in binary form. If NULL was passed in then the result must be freed by the caller.

Definition at line 55 of file creation.c.

References log_pedantic, PLACER, rand_write(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and STACIE_SALT_LENGTH.

Referenced by auth_login(), and register_data_insert_user().

5.403.2.3 stringer_t* stacie_create_shard (stringer_t * output)

Generates a random shard of precisely 64 bytes, suitable for use with a STACIE realm.

Note:

The master key is combined with the realm label and a random shard value to create the realm key.

Parameters:

output a managed string to receive the output; if passed as NULL, an output buffer will be allocated.

Returns:

A managed string holding the freshly generated shard in binary form. If NULL was passed in then the result must be freed by the caller.

Definition at line 19 of file creation.c.

References log_pedantic, PLACER, rand_write(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and STACIE_SHARD_LENGTH.

Referenced by meta_update_realms(), and register_data_insert_user().

5.404 src/providers/stacie/passwords.c File Reference

Process passwords into STACIE key values. `#include "magma.h"`

Functions

- `uint32_t stacie_derive_rounds (stringer_t *password, uint32_t bonus)`
Calculate the number of hash rounds needed for the seed and key derivation stages.
- `stringer_t * stacie_derive_seed (uint32_t rounds, stringer_t *password, stringer_t *salt)`
Concentrates and then extracts the random entropy provided by the password into a seed value for the first hash stage.
- `stringer_t * stacie_derive_key (stringer_t *base, uint32_t rounds, stringer_t *username, stringer_t *password, stringer_t *salt)`
Derive the master key and password key values using the user credentials. The base value provided is dependent upon which of the two keys is being derived. The result is obtained by applying the designated hash function over the input values the appropriate number of times.

5.404.1 Detailed Description

Process passwords into STACIE key values.

Definition in file [passwords.c](#).

5.404.2 Function Documentation

5.404.2.1 `stringer_t* stacie_derive_key (stringer_t * base, uint32_t rounds, stringer_t * username, stringer_t * password, stringer_t * salt)`

Derive the master key and password key values using the user credentials. The base value provided is dependent upon which of the two keys is being derived. The result is obtained by applying the designated hash function over the input values the appropriate number of times.

Note:

The "master key" and the "password key" are referred to as "keys" because deriving them values requires knowing "secret" information, or more specifically, the plain text password and the number of hash rounds being applied. The number of hash rounds is considered a "secret" because it requires knowledge of the plain text password to derive. If an attacker were able to discover the number of rounds they could, at least in theory, use that information to derive the length of a password. If you don't understand why an attacker knowing the length of a password might be considered a bad thing, then I pity you.

Parameters:

- base** The initial entropy being applied to the key derivation process. For the master key derivation, this should be the entropy seed value. For the password key derivation, this should be the master key value. Because the base value is a created using the designated hash function, its length must match output size. For Magma, which is using SHA-512, that means the base value must always be 64 bytes.
- username** The pre-processed username, encoded in UTF8. The rules for normalization and equivalence are different for every domain/deployment. Depending on those rules, the username passed in may be just the local part, or it may consist of the full email address, which would includes the separator (usually the '@' symbol) and a domain name (preferably a fully qualified domain).
- password** The pre-processed password, encoded in UTF8 and normalized using the canonical composition form (aka NFC).
- salt** The user specific salt, which provides protection against precomputed lookup tables. For Magma, the salt value must always be 128 bytes in length;

See also:

[stacie_derive_seed\(\)](#) for more on the salt [length](#).

Returns:

provides a managed string with the derived key stored in a secure memory buffer, or NULL if an error occurs. The length of the output depends on the hash function being used. Magma currently uses SHA-512, which will result in the output being exactly 64 bytes, every single time.

Definition at line 183 of file passwords.c.

References CONTIGUOUS, count, EVP_DigestFinal_d, EVP_DigestInit_ex_d, EVP_DigestUpdate_d, EVP_MD_CTX_cleanup_d, EVP_MD_CTX_init_d, EVP_sha512_d, log_error, log_pedantic, MANAGED_T, MEMORYBUF, SECURE, ssl_error_string(), st_alloc_opts(), st_char_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_int(), st_length_set(), STACIE_KEY_ROUNDS_MAX, STACIE_SALT_LENGTH, and status.

Referenced by auth_stacie().

5.404.2.2 uint32_t stacie_derive_rounds (stringer_t * password, uint32_t bonus)

Calculate the number of hash rounds needed for the seed and key derivation stages.

Note:

This function has an effective maximum of 16,777,216 because STACIE requires the current round be appended to the string being hashed. This value is supplied as an unsigned 24 bit big endian integer. The prescribed maximum is derived from the fact that the total number of possible values for an unsigned 24 bit integer is 16,777,216. This includes 0, since the values being appended by the derivation functions start at 0.

Remarks:

While on the subject of a maximum value, if 16,777,216 hash rounds doesn't provide sufficient protection then the the problem is likely the password, or the hash function, and not the number of rounds.

Parameters:

password A password which may contain any valid Unicode character (presumably encoded using UTF-8).

bonus The number of additional rounds which should be added beyond the number of dynamic rounds calculated using the password length.

Returns:

Valid passwords will return a value between 8 (the prescribed minimum) and 16,777,216 (the prescribed maximum). If an error occurs, then 0 will be returned.

Definition at line 28 of file passwords.c.

References log_pedantic, STACIE_KEY_ROUNDS_MAX, STACIE_KEY_ROUNDS_MIN, uint64_clamp(), and utf8_length_st().

Referenced by auth_stacie().

5.404.2.3 stringer_t* stacie_derive_seed (uint32_t rounds, stringer_t * password, stringer_t * salt)

Concentrates and then extracts the random entropy provided by the password into a seed value for the first hash stage. [passwords.c](#)

Note:

This implementation requires a seed value of 128 bytes because it was written specifically for Magma. However, the STACIE specification does technically allow the salt value to be empty, or of a different length, and provides additional logic in those circumstances. Those additional rules have not been implemented, so if this implementation does not receive a salt value that is exactly 128 bytes in length, it will return NULL to indicate the error.

Parameters:

rounds The derived number of rounds, based on the password length, bonus rounds, and applicable limits.

password The pre-processed password, encoded in UTF8 and normalized using the canonical composition form (aka NFC).

salt The user specific salt, which provides protection against precomputed lookup tables. The salt must be 128 bytes in length.

Returns:

provides a managed string with the entropy seed value stored in a secure memory buffer, or NULL if an error occurs. The length of the return value depends on the HMAC function being used. Magma currently uses an HMAC function based around SHA-512, which results in the output being exactly 64 bytes, at least it was 64 bytes every time I bothered to look.

Definition at line 81 of file passwords.c.

References CONTIGUOUS, count, EVP_sha512_d, HMAC_CTX_cleanup_d, HMAC_CTX_init_d, HMAC_Final_d, HMAC_Init_ex_d, HMAC_Update_d, log_error, log_pedantic, MANAGED_T, MEMORYBUF, SECURE, ssl_error_string(), st_alloc_opts(), st_cleanup, st_data_get(), st_empty_out(), st_length_set(), STACIE_KEY_ROUNDS_MAX, and STACIE_SALT_LENGTH.

Referenced by auth_stacie().

5.405 src/providers/stacie/realms.c File Reference

Service STACIE realm key and vector shard values. `#include "magma.h"`

Functions

- `stringer_t * stacie_realms_key (stringer_t *master_key, stringer_t *realm, stringer_t *salt, stringer_t *shard)`
Derive the symmetric encryption keys applicable to a given realm.
- `stringer_t * stacie_realms_cipher (stringer_t *realm_key)`
Extract the symmetric encryption key from the realm key.
- `stringer_t * stacie_realms_vector (stringer_t *realm_key)`
Extract the vector key from the realm key.
- `stringer_t * stacie_realms_tag (stringer_t *realm_key)`
Extract the tag key from the realm key.

5.405.1 Detailed Description

Service STACIE realm key and vector shard values.

Definition in file [realms.c](#).

5.405.2 Function Documentation

5.405.2.1 `stringer_t* stacie_realms_cipher (stringer_t * realm_key)`

Extract the symmetric encryption key from the realm key. [realms.c](#)

Parameters:

realm_key The complete realm key, which holds the vector, tag and cipher key values.

Returns:

provides a managed string with the symmetric encryption key stored in a secure memory buffer, or NULL if an error occurs.

Definition at line 126 of file [realms.c](#).

References [CONTIGUOUS](#), [log_error](#), [MANAGED_T](#), [PLACER](#), [SECURE](#), [st_data_get\(\)](#), [st_dup_opts\(\)](#), [st_empty](#), and [st_length_get\(\)](#).

5.405.2.2 `stringer_t* stacie_realms_key (stringer_t * master_key, stringer_t * realm, stringer_t * salt, stringer_t * shard)`

Derive the symmetric encryption keys applicable to a given realm.

Note:

It is important that implementations use unique shard's for every realm. This ensures that if a realm key is ever compromised, the data in alternate realms will remain secure. Also, depending on the implementation, the shard might only be stored on the server. This means if the client device is ever lost, a user can protect the information on the device by changing their password. This ensures that any data residing on the lost device will remain protected, even if the lost device holds the plain text of the former password or the stale master key. This operational methodology won't work for everyone, as it also prevents data from being accessible offline.

Parameters:

- master_key* The master key, which was derived using the user's password.
- realm* Realm human readable name for the category of data protected using the given symmetric key.
- salt* The user specific salt value. For legacy derivations, this will be a shard duplicate.
- shard* The shard is the realm specific portion of the key. The shard must be exactly 64 bytes in length.

Returns:

provides a managed string with the realm specific key stored in a secure memory buffer, or NULL if an error occurs. The length of the output depends on the hash function being used. Magma currently uses SHA-512, which will result in the output being exactly 64 bytes. The output will contain the cipher key, and the initialization vector, which will need to be parsed out.

Definition at line 30 of file realms.c.

References CONTIGUOUS, EVP_DigestFinal_d, EVP_DigestInit_ex_d, EVP_DigestUpdate_d, EVP_MD_CTX_cleanup_d, EVP_MD_CTX_init_d, EVP_sha512_d, log_error, log_pedantic, MANAGED_T, MEMORYBUF, PLACER, SECURE, ssl_error_string(), st_alloc_opts(), st_cleanup, st_empty_out(), st_xor(), and STACIE_KEY_LENGTH.

Referenced by meta_update_realms(), and register_data_insert_user().

5.405.2.3 stringer_t* stacie_realm_tag (stringer_t * realm_key)

Extract the tag key from the realm key.

Parameters:

- realm_key* The complete realm key, which holds the vector, tag and cipher key values.

Returns:

provides a managed string with the tag key stored in a secure memory buffer, or NULL if an error occurs.

Definition at line 170 of file realms.c.

References CONTIGUOUS, log_error, MANAGED_T, PLACER, SECURE, st_data_get(), st_dupe_opts(), st_empty, and st_length_get().

5.405.2.4 stringer_t* stacie_realm_vector (stringer_t * realm_key)

Extract the vector key from the realm key.

Parameters:

- realm_key* The complete realm key, which holds the vector, tag and cipher key values.

Returns:

provides a managed string with the vector key stored in a secure memory buffer, or NULL if an error occurs.

Definition at line 148 of file realms.c.

References CONTIGUOUS, log_error, MANAGED_T, PLACER, SECURE, st_data_get(), st_dupe_opts(), st_empty, and st_length_get().

5.406 src/providers/stacie/stacie.h File Reference

These functions implement the Safely Turning Authentication Credentials Into Entropy (STACIE) standard. This standard defines how process passwords into encryption keys, and/or authentication tokens, derive realm specific encryption keys, and perform symmetric encryption using the realm keys. The inputs passed into these functions must be santized, and normalized to ensure a deterministic output.

Defines

- #define [STACIE_KEY_ROUNDS_MIN](#) 8
- #define [STACIE_KEY_ROUNDS_MAX](#) 16777216
- #define [STACIE_TOKEN_ROUNDS](#) 8
- #define [STACIE_SALT_LENGTH](#) 128
- #define [STACIE_NONCE_LENGTH](#) 128
- #define [STACIE_KEY_LENGTH](#) 64
- #define [STACIE_TOKEN_LENGTH](#) 64
- #define [STACIE_SHARD_LENGTH](#) 64
- #define [STACIE_ENCRYPT_MIN](#) 1
- #define [STACIE_ENCRYPT_MAX](#) 16777215
- #define [STACIE_BLOCK_LENGTH](#) 16
- #define [STACIE_ENVELOPE_LENGTH](#) 34

Functions

- [stringer_t * stacie_realm_cipher](#) ([stringer_t](#) *realm_key)
realms.c
- [stringer_t * stacie_realm_key](#) ([stringer_t](#) *master_key, [stringer_t](#) *realm, [stringer_t](#) *salt, [stringer_t](#) *shard)
Derive the symmetric encryption keys applicable to a given realm.
- [stringer_t * stacie_realm_tag](#) ([stringer_t](#) *realm_key)
Extract the tag key from the realm key.
- [stringer_t * stacie_realm_vector](#) ([stringer_t](#) *realm_key)
Extract the vector key from the realm key.
- [stringer_t * stacie_create_nonce](#) ([stringer_t](#) *output)
creation.c
- [stringer_t * stacie_create_salt](#) ([stringer_t](#) *output)
Generates a random salt of precisely 128 bytes, suitable for use with STACIE authentication scheme.
- [stringer_t * stacie_create_shard](#) ([stringer_t](#) *output)
Generates a random shard of precisely 64 bytes, suitable for use with a STACIE realm.
- [stringer_t * stacie_derive_seed](#) (uint32_t rounds, [stringer_t](#) *password, [stringer_t](#) *salt)
passwords.c
- [stringer_t * stacie_derive_key](#) ([stringer_t](#) *base, uint32_t rounds, [stringer_t](#) *username, [stringer_t](#) *password, [stringer_t](#) *salt)
Derive the master key and password key values using the user credentials. The base value provided is dependent upon which of the two keys is being derived. The result is obtained by applying the designated hash function over the input values the appropriate number of times.
- uint32_t [stacie_derive_rounds](#) ([stringer_t](#) *password, uint32_t bonus)

Calculate the number of hash rounds needed for the seed and key derivation stages.

- `stringer_t * stacie_derive_token (stringer_t *base, stringer_t *username, stringer_t *salt, stringer_t *nonce)`
tokens.c
- `stringer_t * stacie_decrypt (stringer_t *vector_key, stringer_t *tag_key, stringer_t *cipher_key, stringer_t *buffer)`
crypto.c
- `stringer_t * stacie_encrypt (uint16_t serial, stringer_t *vector_key, stringer_t *tag_key, stringer_t *cipher_key, stringer_t *buffer)`

5.406.1 Detailed Description

These functions implement the Safely Turning Authentication Credentials Into Entropy (STACIE) standard. This standard defines how process passwords into encryption keys, and/or authentication tokens, derive realm specific encryption keys, and perform symmetric encryption using the realm keys. The inputs passed into these functions must be santized, and normalized to ensure a deterministic output.

See also:

<https://tools.ietf.org/html/draft-ladar-stacie>

Definition in file [stacie.h](#).

5.406.2 Define Documentation

5.406.2.1 #define STACIE_BLOCK_LENGTH 16

Definition at line 36 of file [stacie.h](#).

Referenced by [stacie_encrypt\(\)](#).

5.406.2.2 #define STACIE_ENCRYPT_MAX 16777215

Definition at line 35 of file [stacie.h](#).

Referenced by [stacie_encrypt\(\)](#).

5.406.2.3 #define STACIE_ENCRYPT_MIN 1

Definition at line 34 of file [stacie.h](#).

5.406.2.4 #define STACIE_ENVELOPE_LENGTH 34

Definition at line 37 of file [stacie.h](#).

Referenced by [stacie_decrypt\(\)](#), and [stacie_encrypt\(\)](#).

5.406.2.5 #define STACIE_KEY_LENGTH 64

Definition at line 29 of file [stacie.h](#).

Referenced by [meta_update_realms\(\)](#), [prime_key_decrypt\(\)](#), and [stacie_realms_key\(\)](#).

5.406.2.6 **#define STACIE_KEY_ROUNDS_MAX 16777216**

Definition at line 19 of file stacie.h.

Referenced by `auth_data_fetch()`, `auth_data_update_legacy()`, `auth_stacie()`, `stacie_derive_key()`, `stacie_derive_rounds()`, and `stacie_derive_seed()`.

5.406.2.7 **#define STACIE_KEY_ROUNDS_MIN 8**

Definition at line 18 of file stacie.h.

Referenced by `stacie_derive_rounds()`.

5.406.2.8 **#define STACIE_NONCE_LENGTH 128**

Definition at line 26 of file stacie.h.

Referenced by `auth_challenge()`, `auth_response()`, `stacie_create_nonce()`, and `stacie_derive_token()`.

5.406.2.9 **#define STACIE_SALT_LENGTH 128**

Definition at line 25 of file stacie.h.

Referenced by `auth_data_fetch()`, `stacie_create_salt()`, `stacie_derive_key()`, `stacie_derive_seed()`, and `stacie_derive_token()`.

5.406.2.10 **#define STACIE_SHARD_LENGTH 64**

Definition at line 31 of file stacie.h.

Referenced by `meta_data_fetch_shard()`, `meta_data_insert_shard()`, `meta_update_realms()`, and `stacie_create_shard()`.

5.406.2.11 **#define STACIE_TOKEN_LENGTH 64**

Definition at line 30 of file stacie.h.

Referenced by `auth_data_fetch()`, and `auth_response()`.

5.406.2.12 **#define STACIE_TOKEN_ROUNDS 8**

Definition at line 22 of file stacie.h.

Referenced by `stacie_derive_token()`.

5.406.3 Function Documentation

5.406.3.1 **stringer_t* stacie_create_nonce (stringer_t * *output*)**

[creation.c](#) [creation.c](#)

Note:

While the salt and nonce creation functions are nearly identical, they remain separated so they can use distinct preprocessor definitions for the result length.

Parameters:

output a managed string to receive the output; if passed as NULL, an output buffer will be allocated.

Returns:

A managed string holding the freshly generated nonce in binary form. If NULL was passed in then the result must be freed by the caller.

Definition at line 91 of file creation.c.

References log_pedantic, PLACER, rand_write(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and STACIE_NONCE_LENGTH.

Referenced by auth_challenge(), and auth_response().

5.406.3.2 stringer_t* stacie_create_salt (stringer_t * output)

Generates a random salt of precisely 128 bytes, suitable for use with STACIE authentication scheme.

Note:

While the salt and nonce creation functions are nearly identical, they remain separated so they can use distinct preprocessor definitions for the result length.

Parameters:

output a managed string to receive the output; if passed as NULL, an output buffer will be allocated.

Returns:

A managed string holding the freshly generated salt in binary form. If NULL was passed in then the result must be freed by the caller.

Definition at line 55 of file creation.c.

References log_pedantic, PLACER, rand_write(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and STACIE_SALT_LENGTH.

Referenced by auth_login(), and register_data_insert_user().

5.406.3.3 stringer_t* stacie_create_shard (stringer_t * output)

Generates a random shard of precisely 64 bytes, suitable for use with a STACIE realm.

Note:

The master key is combined with the realm label and a random shard value to create the realm key.

Parameters:

output a managed string to receive the output; if passed as NULL, an output buffer will be allocated.

Returns:

A managed string holding the freshly generated shard in binary form. If NULL was passed in then the result must be freed by the caller.

Definition at line 19 of file creation.c.

References log_pedantic, PLACER, rand_write(), st_alloc(), st_avail_get(), st_cleanup, st_data_get(), st_length_set(), st_opt_get(), st_valid_destination(), st_valid_tracked(), and STACIE_SHARD_LENGTH.

Referenced by meta_update_realms(), and register_data_insert_user().

5.406.3.4 stringer_t* stacie_decrypt (stringer_t * vector_key, stringer_t * tag_key, stringer_t * cipher_key, stringer_t * buffer)

crypto.c

Parameters:

vector_key
tag_key
cipher_key
buffer

Returns:

Definition at line 185 of file crypto.c.

References `EVP_aes_256_gcm_d`, `EVP_CIPHER_CTX_cleanup_d`, `EVP_CIPHER_CTX_ctrl_d`, `EVP_CIPHER_CTX_init_d`, `EVP_CIPHER_CTX_key_length_d`, `EVP_DecryptFinal_ex_d`, `EVP_DecryptInit_ex_d`, `EVP_DecryptUpdate_d`, `log_error`, `log_pedantic`, `MANAGEDBUF`, `MEMORYBUF`, `mm_move()`, `PLACER`, `ssl_error_string()`, `st_alloc()`, `st_data_get()`, `st_empty_out()`, `st_free()`, `st_length_get()`, `st_length_set()`, `st_xor()`, and `STACIE_ENVELOPE_LENGTH`.

5.406.3.5 `stringer_t* stacie_derive_key (stringer_t * base, uint32_t rounds, stringer_t * username, stringer_t * password, stringer_t * salt)`

Derive the master key and password key values using the user credentials. The base value provided is dependent upon which of the two keys is being derived. The result is obtained by applying the designated hash function over the input values the appropriate number of times.

Note:

The "master key" and the "password key" are referred to as "keys" because deriving them values requires knowing "secret" information, or more specifically, the plain text password and the number of hash rounds being applied. The number of hash rounds is considered a "secret" because it requires knowledge of the plain text password to derive. If an attacker were able to discover the number of rounds they could, at least in theory, use that information to derive the length of a password. If you don't understand why an attacker knowing the length of a password might be considered a bad thing, then I pity you.

Parameters:

base The initial entropy being applied to the key derivation process. For the master key derivation, this should be the entropy seed value. For the password key derivation, this should be the master key value. Because the base value is a created using the designated hash function, its length must match output size. For Magma, which is using SHA-512, that means the base value must always be 64 bytes.

username The pre-processed username, encoded in UTF8. The rules for normalization and equivalence are different for every domain/deployment. Depending on those rules, the username passed in may be just the local part, or it may consist of the full email address, which would includes the separator (usually the '@' symbol) and a domain name (preferably a fully qualified domain).

password The pre-processed password, encoded in UTF8 and normalized using the canonical composition form (aka NFC).

salt The user specific salt, which provides protection against precomputed lookup tables. For Magma, the salt value must always be 128 bytes in length;

See also:

[stacie_derive_seed\(\)](#) for more on the salt [length](#).

Returns:

provides a managed string with the derived key stored in a secure memory buffer, or NULL if an error occurs. The length of the output depends on the hash function being used. Magma currently uses SHA-512, which will result in the output being exactly 64 bytes, every single time.

Definition at line 183 of file passwords.c.

References `CONTIGUOUS`, `count`, `EVP_DigestFinal_d`, `EVP_DigestInit_ex_d`, `EVP_DigestUpdate_d`, `EVP_MD_CTX_cleanup_d`, `EVP_MD_CTX_init_d`, `EVP_sha512_d`, `log_error`, `log_pedantic`, `MANAGED_T`, `MEMORYBUF`, `SECURE`, `ssl_error_string()`, `st_alloc_opts()`,

st_char_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_int(), st_length_set(), STACIE_KEY_ROUNDS_MAX, STACIE_SALT_LENGTH, and status.

Referenced by auth_stacie().

5.406.3.6 uint32_t stacie_derive_rounds (stringer_t * password, uint32_t bonus)

Calculate the number of hash rounds needed for the seed and key derivation stages.

Note:

This function has an effective maximum of 16,777,216 because STACIE requires the current round be appended to the string being hashed. This value is supplied as an unsigned 24 bit big endian integer. The prescribed maximum is derived from the fact that the total number of possible values for an unsigned 24 bit integer is 16,777,216. This includes 0, since the values being appended by the derivation functions start at 0.

Remarks:

While on the subject of a maximum value, if 16,777,216 hash rounds doesn't provide sufficient protection then the the problem is likely the password, or the hash function, and not the number of rounds.

Parameters:

password A password which may contain any valid Unicode character (presumably encoded using UTF-8).

bonus The number of additional rounds which should be added beyond the number of dynamic rounds calculated using the password length.

Returns:

Valid passwords will return a value between 8 (the prescribed minimum) and 16,777,216 (the prescribed maximum). If an error occurs, then 0 will be returned.

Definition at line 28 of file passwords.c.

References log_pedantic, STACIE_KEY_ROUNDS_MAX, STACIE_KEY_ROUNDS_MIN, uint64_clamp(), and utf8_length_st().

Referenced by auth_stacie().

5.406.3.7 stringer_t* stacie_derive_seed (uint32_t rounds, stringer_t * password, stringer_t * salt)

[passwords.c](#) [passwords.c](#)

Note:

This implementation requires a seed value of 128 bytes because it was written specifically for Magma. However, the STACIE specification does technically allow the salt value to be empty, or of a different length, and provides additional logic in those circumstances. Those additional rules have not been implemented, so if this implementation does not receive a salt value that is exactly 128 bytes in length, it will return NULL to indicate the error.

Parameters:

rounds The derived number of rounds, based on the password length, bonus rounds, and applicable limits.

password The pre-processed password, encoded in UTF8 and normalized using the canonical composition form (aka NFC).

salt The user specific salt, which provides protection against precomputed lookup tables. The salt must be 128 bytes in length.

Returns:

provides a managed string with the entropy seed value stored in a secure memory buffer, or NULL if an error occurs. The length of the return value depends on the HMAC function being used. Magma currently uses an HMAC function based around SHA-512, which results in the output being exactly 64 bytes, at least it was 64 bytes every time I bothered to look.

Definition at line 81 of file passwords.c.

References CONTIGUOUS, count, EVP_sha512_d, HMAC_CTX_cleanup_d, HMAC_CTX_init_d, HMAC_Final_d, HMAC_Init_ex_d, HMAC_Update_d, log_error, log_pedantic, MANAGED_T, MEMORYBUF, SECURE, ssl_error_string(), st_alloc_opts(), st_cleanup, st_data_get(), st_empty_out(), st_length_set(), STACIE_KEY_ROUNDS_MAX, and STACIE_SALT_LENGTH.

Referenced by auth_stacie().

5.406.3.8 stringer_t* stacie_derive_token (stringer_t * *base*, stringer_t * *username*, stringer_t * *salt*, stringer_t * *nonce*)

[tokens.c](#) [tokens.c](#)

Note:

The token values are used for authentication with an untrusted server, which needs to authenticate a user, but does not know anything about the plain text password. Untrusted servers store the static verification token, and using this function, can verify ephemeral login tokens. Servers can also authenticate a password change request by checking whether the supplied password key, if passed to this the token derivation function, matches the stored verification token. The base value passed in is dependent upon which token is being derived. The result is obtained by applying the designated hash function over the input values a fixed number of times.

Parameters:

- base** The initial entropy being applied to the token derivation process. For the static verification token, this will be the password key value. For an ephemeral login token, this will be the verification token. Because the base value is a created using the designated hash function, its length must match output size. For Magma, which is using SHA-512, that means the base value must always be 64 bytes.
- username** The pre-processed username, encoded in UTF8. The rules for normalization and equivalence are different for every domain/deployment. Depending on those rules, the username passed in may be just the local part, or it may consist of the full email address, which would includes the separator (usually the '@' symbol) and a domain name (preferably a fully qualified domain).
- salt** The user specific salt, which provides protection against precomputed lookup tables. For Magma, the salt value must always be 128 bytes in length;

See also:

[stacie_derive_seed\(\)](#) for more on the salt [length](#).

Parameters:

- nonce** When deriving an ephemeral login token, an additional single use, randomly generated value is supplied as the nonce parameter. For Magma, this value must always be 128 bytes in length.

Returns:

provides a managed string with the derived token stored in a secure memory buffer, or NULL if an error occurs. The length of the output depends on the hash function being used. Magma currently uses SHA-512, which will result in the output being exactly 64 bytes, with a level of reliability that rivals death and taxes.

Definition at line 38 of file tokens.c.

References CONTIGUOUS, count, EVP_DigestFinal_d, EVP_DigestInit_ex_d, EVP_DigestUpdate_d, EVP_MD_CTX_cleanup_d, EVP_MD_CTX_init_d, EVP_sha512_d, log_error, log_pedantic, MANAGED_T, MEMORYBUF, SECURE, ssl_error_string(), st_alloc_opts(), st_char_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_int(), st_length_set(), STACIE_NONCE_LENGTH, STACIE_SALT_LENGTH, and STACIE_TOKEN_ROUNDS.

Referenced by auth_stacie().

5.406.3.9 stringer_t* stacie_encrypt (uint16_t *serial*, stringer_t * *vector_key*, stringer_t * *tag_key*, stringer_t * *cipher_key*, stringer_t * *buffer*)

Parameters:

serial

*vector_key**tag_key**cipher_key**buffer***Returns:**

Definition at line 21 of file crypto.c.

References `EVP_aes_256_gcm_d`, `EVP_CIPHER_CTX_cleanup_d`, `EVP_CIPHER_CTX_ctrl_d`, `EVP_CIPHER_CTX_init_d`, `EVP_CIPHER_CTX_key_length_d`, `EVP_EncryptFinal_ex_d`, `EVP_EncryptInit_ex_d`, `EVP_EncryptUpdate_d`, `log_error`, `log_pedantic`, `MANAGEDBUF`, `mm_move()`, `PLACER`, `rand_write()`, `st_alloc()`, `st_data_get()`, `st_empty_out()`, `st_free()`, `st_length_get()`, `st_length_set()`, `st_xor()`, `STACIE_BLOCK_LENGTH`, `STACIE_ENCRYPT_MAX`, and `STACIE_ENVELOPE_LENGTH`.

5.406.3.10 stringer_t* stacie_realm_cipher (stringer_t * *realm_key*)

[realms.c](#) [realms.c](#)

Parameters:

realm_key The complete realm key, which holds the vector, tag and cipher key values.

Returns:

provides a managed string with the symmetric encryption key stored in a secure memory buffer, or NULL if an error occurs.

Definition at line 126 of file realms.c.

References `CONTIGUOUS`, `log_error`, `MANAGED_T`, `PLACER`, `SECURE`, `st_data_get()`, `st_dupe_opts()`, `st_empty`, and `st_length_get()`.

5.406.3.11 stringer_t* stacie_realm_key (stringer_t * *master_key*, stringer_t * *realm*, stringer_t * *salt*, stringer_t * *shard*)

Derive the symmetric encryption keys applicable to a given realm.

Note:

It is important that implementations use unique shard's for every realm. This ensures that if a realm key is ever compromised, the data in alternate realms will remain secure. Also, depending on the implementation, the shard might only be stored on the server. This means if the client device is ever lost, a user can protect the information on the device by changing their password. This ensures that any data residing on the lost device will remain protected, even if the lost device holds the plain text of the former password or the stale master key. This operational methodology won't work for everyone, as it also prevents data from being accessible offline.

Parameters:

master_key The master key, which was derived using the user's password.

realm Realm human readable name for the category of data protected using the given symmetric key.

salt The user specific salt value. For legacy derivations, this will be a shard duplicate.

shard The shard is the realm specific portion of the key. The shard must be exactly 64 bytes in length.

Returns:

provides a managed string with the realm specific key stored in a secure memory buffer, or NULL if an error occurs. The length of the output depends on the hash function being used. Magma currently uses SHA-512, which will result in the output being exactly 64 bytes. The output will contain the cipher key, and the initialization vector, which will need to be parsed out.

Definition at line 30 of file realms.c.

References CONTIGUOUS, EVP_DigestFinal_d, EVP_DigestInit_ex_d, EVP_DigestUpdate_d, EVP_MD_CTX_cleanup_d, EVP_MD_CTX_init_d, EVP_sha512_d, log_error, log_pedantic, MANAGED_T, MEMORYBUF, PLACER, SECURE, ssl_error_string(), st_alloc_opts(), st_cleanup, st_empty_out(), st_xor(), and STACIE_KEY_LENGTH.

Referenced by meta_update_realms(), and register_data_insert_user().

5.406.3.12 stringer_t* stacie_realm_tag (stringer_t * *realm_key*)

Extract the tag key from the realm key.

Parameters:

realm_key The complete realm key, which holds the vector, tag and cipher key values.

Returns:

provides a managed string with the tag key stored in a secure memory buffer, or NULL if an error occurs.

Definition at line 170 of file realms.c.

References CONTIGUOUS, log_error, MANAGED_T, PLACER, SECURE, st_data_get(), st_dupe_opts(), st_empty, and st_length_get().

5.406.3.13 stringer_t* stacie_realm_vector (stringer_t * *realm_key*)

Extract the vector key from the realm key.

Parameters:

realm_key The complete realm key, which holds the vector, tag and cipher key values.

Returns:

provides a managed string with the vector key stored in a secure memory buffer, or NULL if an error occurs.

Definition at line 148 of file realms.c.

References CONTIGUOUS, log_error, MANAGED_T, PLACER, SECURE, st_data_get(), st_dupe_opts(), st_empty, and st_length_get().

5.407 src/providers/stacie/tokens.c File Reference

Derive STACIE token values. `#include "magma.h"`

Functions

- `stringer_t * stacie_derive_token (stringer_t *base, stringer_t *username, stringer_t *salt, stringer_t *nonce)`
Derive the token values using the supplied input values. The base value supplied depends on which token is being derived.

5.407.1 Detailed Description

Derive STACIE token values.

Definition in file [tokens.c](#).

5.407.2 Function Documentation

5.407.2.1 `stringer_t* stacie_derive_token (stringer_t *base, stringer_t *username, stringer_t *salt, stringer_t *nonce)`

Derive the token values using the supplied input values. The base value supplied depends on which token is being derived. [tokens.c](#)

Note:

The token values are used for authentication with an untrusted server, which needs to authenticate a user, but does not know anything about the plain text password. Untrusted servers store the static verification token, and using this function, can verify ephemeral login tokens. Servers can also authenticate a password change request by checking whether the supplied password key, if passed to this the token derivation function, matches the stored verification token. The base value passed in is dependent upon which token is being derived. The result is obtained by applying the designated hash function over the input values a fixed number of times.

Parameters:

- base** The initial entropy being applied to the token derivation process. For the static verification token, this will be the password key value. For an ephemeral login token, this will be the verification token. Because the base value is a created using the designated hash function, its length must match output size. For Magma, which is using SHA-512, that means the base value must always be 64 bytes.
- username** The pre-processed username, encoded in UTF8. The rules for normalization and equivalence are different for every domain/deployment. Depending on those rules, the username passed in may be just the local part, or it may consist of the full email address, which would includes the separator (usually the '@' symbol) and a domain name (preferably a fully qualified domain).
- salt** The user specific salt, which provides protection against precomputed lookup tables. For Magma, the salt value must always be 128 bytes in length;

See also:

[stacie_derive_seed\(\)](#) for more on the salt [length](#).

Parameters:

- nonce** When deriving an ephemeral login token, an additional single use, randomly generated value is supplied as the nonce parameter. For Magma, this value must always be 128 bytes in length.

Returns:

provides a managed string with the derived token stored in a secure memory buffer, or NULL if an error occurs. The length of the output depends on the hash function being used. Magma currently uses SHA-512, which will result in the output being exactly 64 bytes, with a level of reliability that rivals death and taxes.

Definition at line 38 of file tokens.c.

References CONTIGUOUS, count, EVP_DigestFinal_d, EVP_DigestInit_ex_d, EVP_DigestUpdate_d, EVP_MD_CTX_cleanup_d, EVP_MD_CTX_init_d, EVP_sha512_d, log_error, log_pedantic, MANAGED_T, MEMORYBUF, SECURE, ssl_error_string(), st_alloc_opts(), st_char_get(), st_cleanup, st_data_get(), st_empty_out(), st_length_int(), st_length_set(), STACIE_NONCE_LENGTH, STACIE_SALT_LENGTH, and STACIE_TOKEN_ROUNDS.

Referenced by auth_stacie().

5.408 src/providers/storage/storage.h File Reference

Defines

- #define [TANK_ENTRY_VERSION](#) 100
- #define [TANK_RECORD_VERSION](#) 100

Enumerations

- enum { [TANK_COMPRESS_LZO](#) = 1, [TANK_COMPRESS_ZLIB](#) = 2, [TANK_COMPRESS_BZIP](#) = 4 }

Functions

- struct [__attribute__](#) ((packed))
- [bool_t lib_load_tokyo](#) (void)
Initialize the Tokyo Cabinet library and bind dynamically to the exported functions that are required.
- const [chr_t](#) * [lib_version_tokyo](#) (void)
- [uint64_t tree_count](#) (void *inx)
Binary trees.
- [inx_t](#) * [tree_alloc](#) ([uint64_t](#) options, void *data_free)
Create a new on-memory tree database.
- void [tank_stop](#) (void)
Startup and shutdown.
- [bool_t tank_start](#) (void)
Initialize the local tank storage system.
- [uint64_t tank_size](#) (void)
Info functions.
- [uint64_t tank_count](#) (void)
Count the number of objects in all local storage tanks.
- [uint64_t tank_cycle](#) (void)
Get the next storage tank.
- void [tank_maintain](#) (void)
Maintenance.
- [bool_t tank_delete](#) ([uint64_t](#) hnum, [uint64_t](#) tnum, [uint64_t](#) unum, [uint64_t](#) onum)
Object handling.
- [stringer_t](#) * [tank_load](#) ([uint64_t](#) hnum, [uint64_t](#) tnum, [uint64_t](#) unum, [uint64_t](#) onum)
- [uint64_t tank_store](#) ([uint64_t](#) hnum, [uint64_t](#) tnum, [uint64_t](#) unum, [stringer_t](#) *data, [uint64_t](#) flags)
Store a binary object on the file system.
- [bool_t tank_delete_object](#) ([int64_t](#) transaction, [uint64_t](#) hnum, [uint64_t](#) tnum, [uint64_t](#) unum, [uint64_t](#) onum)
Delete a tank object from the mysql database.

- uint64_t [tank_insert_object](#) (int64_t transaction, uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t size, uint64_t flags)
Insert a tank object into the mysql database.

Variables

- enum { ... } [TANK_FLAGS_E](#)
- [record_t](#)
- [entry_t](#)

5.408.1 Define Documentation

5.408.1.1 #define TANK_ENTRY_VERSION 100

Definition at line 18 of file storage.h.

Referenced by [tank_store\(\)](#).

5.408.1.2 #define TANK_RECORD_VERSION 100

Definition at line 19 of file storage.h.

Referenced by [tank_load\(\)](#), and [tank_store\(\)](#).

5.408.2 Enumeration Type Documentation

5.408.2.1 anonymous enum

Enumerator:

TANK_COMPRESS_LZO

TANK_COMPRESS_ZLIB

TANK_COMPRESS_BZIP

Definition at line 21 of file storage.h.

5.408.3 Function Documentation

5.408.3.1 struct __attribute__((packed)) [read]

< Number indicating the entry version, which also tells us the layout of the data.

< The length of the record data.

< A collection of bit mask flags to indicate whether the object was compressed, encrypted, or replicated.

< Which local storage tank was used to store the object.

< The user number of the object owner.

< The object number.

< The serial number. Starts at zero, and increments for each update.

< Time stamp taken when the object is stored.

< The full length of the original data.

- < The compressed length of the data, if applicable.
- < The length of the encrypted data block, if applicable.
- < Number indicating the entry version, which also tells us the layout of the data.
- < Which local storage tank was used to store the object.
- < The user number of the object owner.
- < The object number.
- < The serial number. Starts at zero, and increments for each update.
- < Time stamp taken when the object is stored.
- < When the object was first created.
- < When the object was last updated.
- < When the object was flagged for deletion.
- < When archived/deleted objects can be permanently purged.

Definition at line 49 of file storage.h.

5.408.3.2 **bool_t lib_load_tokyo (void)**

Initialize the Tokyo Cabinet library and bind dynamically to the exported functions that are required.

Returns:

true on success or false on failure.

Definition at line 19 of file tokyo.c.

References lib_symbols(), M_BIND, and symbol_t.

Referenced by lib_load().

5.408.3.3 **const chr_t* lib_version_tokyo (void)**

Definition at line 11 of file tokyo.c.

References tcversion_d.

Referenced by lib_load().

5.408.3.4 **uint64_t tank_count (void)**

Count the number of objects in all local storage tanks.

Returns:

the total number of all objects contained in all tanks.

Definition at line 46 of file tank.c.

References count, log_pedantic, store, tanks_num, and tchdbnum_d.

5.408.3.5 **uint64_t tank_cycle (void)**

Get the next storage tank.

Note:

This function cycles through all the storage tanks in order, to ensure homogeneous data distribution.

Returns:

the next storage tank in line.

Definition at line 28 of file tank.c.

References mutex_lock(), mutex_unlock(), tanks_lock, tanks_next, and tanks_num.

5.408.3.6 bool_t tank_delete (uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t onum)

Object handling. Delete a binary object on the file system.

Parameters:

hnum The host number.

tnum The tank number.

unum The user number.

onum The object number.

Returns:

Returns true if the object was deleted without an error.

Definition at line 89 of file tank.c.

References log_critical, log_error, store, tank_delete_object(), tchdbecode_d, tchdberrmsg_d, tchdbout_d, tran_commit(), tran_rollback(), and tran_start().

5.408.3.7 bool_t tank_delete_object (int64_t transaction, uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t onum)

Delete a tank object from the mysql database.

Parameters:

transaction a mysql connection identifier for which a transaction has been started.

hnum the host number.

tnum the storage tank number.

unum the usernum of the user that owns the object.

onum the stored object id.

Returns:

false on failure, or true on success.

Definition at line 67 of file data.c.

References log_pedantic, mm_wipe(), and stmt_exec_affected_conn().

Referenced by tank_delete().

5.408.3.8 `uint64_t tank_insert_object (int64_t transaction, uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t size, uint64_t flags)`

Insert a tank object into the mysql database.

Parameters:

transaction a mysql connection identifier for which a transaction has been started.

hnum the host number.

tnum the storage tank number.

unum the usernum of the user that owns the object.

size the length, in bytes, of the object to be stored.

flags 0 on failure, or the objectnum field for the newly inserted object.

Definition at line 19 of file data.c.

References `mm_wipe()`, and `stmt_insert_conn()`.

Referenced by `tank_store()`.

5.408.3.9 `stringer_t* tank_load (uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t onum)`

Load and decompress the data for the object described by the input parameters.

Parameters:

hnum The host number.

tnum The tank number.

unum The user number.

onum The object number.

Returns:

Returns the object data inside a `stringer_t` or NULL to indicate an error occurred.

Definition at line 150 of file tank.c.

References `decompress_bzip()`, `decompress_lzo()`, `decompress_zlib()`, `log_check`, `log_error`, `log_pedantic`, `mm_copy()`, `record_t`, `st_import()`, `store`, `TANK_COMPRESS_BZIP`, `TANK_COMPRESS_LZO`, `TANK_COMPRESS_ZLIB`, `TANK_RECORD_VERSION`, `tcfree_d`, `tchdbdecode_d`, `tchdberrmsg_d`, and `tchdbget_d`.

5.408.3.10 `void tank_maintain (void)`

Maintenance. Maintenance.

Returns:

This function returns no value.

Definition at line 417 of file tank.c.

References `store`, `tanks_num`, and `tchdbdefrag_d`.

5.408.3.11 uint64_t tank_size (void)

Info functions. Info functions.

Returns:

the total number of bytes occupied by all objects in all storage tanks.

Definition at line 68 of file tank.c.

References store, tanks_num, and tchdbfsiz_d.

5.408.3.12 bool_t tank_start (void)

Initialize the local tank storage system.

Returns:

Returns true if all of the storage tanks were setup successfully and the system is ready to store data.

Definition at line 497 of file tank.c.

References log_critical, magma, MAGMA_FILEPATH_MAX, mm_alloc(), ns_length_get(), magma_t::storage, store, magma_t::tank, tank_open(), and tanks_num.

Referenced by process_start().

5.408.3.13 void tank_stop (void)

Startup and shutdown. Close all of the storage system file handles and release the resources they were using.

Definition at line 532 of file tank.c.

References mm_free(), store, tank_close(), and tanks_num.

Referenced by process_stop().

5.408.3.14 uint64_t tank_store (uint64_t hnum, uint64_t tnum, uint64_t unum, stringer_t * data, uint64_t flags)

Store a binary object on the file system.

Note:

Flags include (TANK_COMPRESS_LZO | TANK_COMPRESS_ZLIB | TANK_COMPRESS_BZIP),

Parameters:

hnum the host number.

tnum the tank number.

unum the userid number.

data a managed string (placer) with the data to be stored.

flags a bitmask of flags specifying encryption, compression, and replication.

Returns:

0 on failure, If the object is stored the object number is returned, or 0 if an error occurs.

Definition at line 256 of file tank.c.

References compress_body_length(), compress_bzip(), compress_free(), compress_lzo(), compress_total_length(), compress_zlib(), entry_t, log_error, mm_alloc(), mm_copy(), mm_free(), mm_wipe(), record_t, st_data_get(), st_length_get(), store, TANK_COMPRESS_BZIP, TANK_COMPRESS_LZO, TANK_COMPRESS_ZLIB, TANK_ENTRY_VERSION, tank_insert_object(), TANK_RECORD_VERSION, tanks_num, tchdbecode_d, tchdberrmsg_d, tchdbout_d, tchdbputasync_d, tran_commit(), tran_rollback(), and tran_start().

5.408.3.15 `inx_t* tree_alloc(uint64_t options, void *data_free)`

Create a new on-memory tree database.

See also:

tcndbnew2()

Parameters:

options

data_free

Returns:

Definition at line 285 of file tree.c.

References inx_t, log_pedantic, mm_alloc(), mm_free(), tcndbnew2_d, tree_cmp(), tree_cursor_alloc(), tree_cursor_free(), tree_cursor_key_active(), tree_cursor_key_next(), tree_cursor_reset(), tree_cursor_value_active(), tree_cursor_value_next(), tree_delete(), tree_find(), tree_free(), tree_insert(), and tree_truncate().

Referenced by inx_alloc().

5.408.3.16 `uint64_t tree_count(void *inx)`

Binary trees. Binary trees.

See also:

tcndbrnum()

Parameters:

inx

Returns:

the record count.

Definition at line 38 of file tree.c.

References inx_t, and tcndbrnum_d.

5.408.4 Variable Documentation

5.408.4.1 `entry_t`

Definition at line 68 of file storage.h.

Referenced by tank_store().

5.408.4.2 record_t

Definition at line 47 of file storage.h.

Referenced by sanity_check(), tank_load(), and tank_store().

5.408.4.3 enum { ... } TANK_FLAGS_E

5.409 src/providers/storage/tank.c File Reference

```
#include "magma.h"
```

Functions

- uint64_t [tank_cycle](#) (void)
Get the next storage tank.
- uint64_t [tank_count](#) (void)
Count the number of objects in all local storage tanks.
- uint64_t [tank_size](#) (void)
Count the amount of space used by all local storage tanks.
- bool_t [tank_delete](#) (uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t onum)
Object handling.
- stringer_t * [tank_load](#) (uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t onum)
- uint64_t [tank_store](#) (uint64_t hnum, uint64_t tnum, uint64_t unum, stringer_t *data, uint64_t flags)
Store a binary object on the file system.
- void [tank_maintain](#) (void)
Perform periodic maintenance on the storage tanks (defragment them in the background).
- TCHDB * [tank_open](#) (char *location)
- void [tank_close](#) (TCHDB *ctx)
- bool_t [tank_start](#) (void)
Initialize the local tank storage system.
- void [tank_stop](#) (void)
Startup and shutdown.

Variables

- uint64_t [tanks_num](#) = 4
- uint64_t [tanks_next](#) = 0
- pthread_mutex_t [tanks_lock](#) = PTHREAD_MUTEX_INITIALIZER
- struct {
 uint8_t [tuner](#)
 TCHDB * [system](#)
 TCHDB ** [tanks](#)
} [store](#)

5.409.1 Function Documentation

5.409.1.1 void tank_close (TCHDB * ctx)

Cleanly closes a storage file context and releases the memory.

Parameters:

ctx The storage context that needs to be closed.

Definition at line 469 of file tank.c.

References `log_error`, `tchdbclose_d`, `tchdbdefrag_d`, `tchdbdel_d`, `tchdbecode_d`, `tchdberrmsg_d`, `tchdboptimize_d`, `tchdbpath_d`, and `tchdbsync_d`.

Referenced by `tank_stop()`.

5.409.1.2 uint64_t tank_count (void)

Count the number of objects in all local storage tanks.

Returns:

the total number of all objects contained in all tanks.

Definition at line 46 of file tank.c.

References `count`, `log_pedantic`, `store`, `tanks_num`, and `tchdbnum_d`.

5.409.1.3 uint64_t tank_cycle (void)

Get the next storage tank.

Note:

This function cycles through all the storage tanks in order, to ensure homogeneous data distribution.

Returns:

the next storage tank in line.

Definition at line 28 of file tank.c.

References `mutex_lock()`, `mutex_unlock()`, `tanks_lock`, `tanks_next`, and `tanks_num`.

5.409.1.4 bool_t tank_delete (uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t onum)

Object handling. Delete a binary object on the file system.

Parameters:

hnum The host number.

tnum The tank number.

unum The user number.

onum The object number.

Returns:

Returns true if the object was deleted without an error.

Definition at line 89 of file tank.c.

References `log_critical`, `log_error`, `store`, `tank_delete_object()`, `tchdbecode_d`, `tchdberrmsg_d`, `tchdbout_d`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

5.409.1.5 stringer_t* tank_load (uint64_t hnum, uint64_t tnum, uint64_t unum, uint64_t onum)

Load and decompress the data for the object described by the input parameters.

Parameters:

hnum The host number.
tnum The tank number.
unum The user number.
onum The object number.

Returns:

Returns the object data inside a stringer_t or NULL to indicate an error occurred.

Definition at line 150 of file tank.c.

References decompress_bzip(), decompress_lzo(), decompress_zlib(), log_check, log_error, log_pedantic, mm_copy(), record_t, st_import(), store, TANK_COMPRESS_BZIP, TANK_COMPRESS_LZO, TANK_COMPRESS_ZLIB, TANK_RECORD_VERSION, tcfree_d, tchdbdecode_d, tchdberrmsg_d, and tchdbget_d.

5.409.1.6 void tank_maintain (void)

Perform periodic maintenance on the storage tanks (defragment them in the background). Maintenance.

Returns:

This function returns no value.

Definition at line 417 of file tank.c.

References store, tanks_num, and tchdbdefrag_d.

5.409.1.7 TCHDB* tank_open (char * location)

Open a storage context for the given location. Use the [store.tuner](#) parameters to configure the context.

Parameters:

location The path to the file where the data is (or will be) stored.

Returns:

Returns a pointer to the storage context on success and NULL on failure.

Definition at line 431 of file tank.c.

References log_critical, tchdbdel_d, tchdbdecode_d, tchdberrmsg_d, tchdbnew_d, tchdbopen_d, tchdbsetmutex_d, and tchdbtune_d.

Referenced by tank_start().

5.409.1.8 uint64_t tank_size (void)

Count the amount of space used by all local storage tanks. Info functions.

Returns:

the total number of bytes occupied by all objects in all storage tanks.

Definition at line 68 of file tank.c.

References store, tanks_num, and tchdbfsiz_d.

5.409.1.9 bool_t tank_start (void)

Initialize the local tank storage system.

Returns:

Returns true if all of the storage tanks were setup successfully and the system is ready to store data.

Definition at line 497 of file tank.c.

References log_critical, magma, MAGMA_FILEPATH_MAX, mm_alloc(), ns_length_get(), magma_t::storage, store, magma_t::tank, tank_open(), and tanks_num.

Referenced by process_start().

5.409.1.10 void tank_stop (void)

Startup and shutdown. Close all of the storage system file handles and release the resources they were using.

Definition at line 532 of file tank.c.

References mm_free(), store, tank_close(), and tanks_num.

Referenced by process_stop().

5.409.1.11 uint64_t tank_store (uint64_t hnum, uint64_t tnum, uint64_t unum, stringer_t * data, uint64_t flags)

Store a binary object on the file system.

Note:

Flags include (TANK_COMPRESS_LZO | TANK_COMPRESS_ZLIB | TANK_COMPRESS_BZIP),

Parameters:

hnum the host number.

tnum the tank number.

unum the userid number.

data a managed string (placer) with the data to be stored.

flags a bitmask of flags specifying encryption, compression, and replication.

Returns:

0 on failure, If the object is stored the object number is returned, or 0 if an error occurs.

Definition at line 256 of file tank.c.

References compress_body_length(), compress_bzip(), compress_free(), compress_lzo(), compress_total_length(), compress_zlib(), entry_t, log_error, mm_alloc(), mm_copy(), mm_free(), mm_wipe(), record_t, st_data_get(), st_length_get(), store, TANK_COMPRESS_BZIP, TANK_COMPRESS_LZO, TANK_COMPRESS_ZLIB, TANK_ENTRY_VERSION, tank_insert_object(), TANK_RECORD_VERSION, tanks_num, tchdbecode_d, tchdberrmsg_d, tchdbout_d, tchdbputasync_d, tran_commit(), tran_rollback(), and tran_start().

5.409.2 Variable Documentation**5.409.2.1 struct { ... } store**

Referenced by _clone_cached_object(), _destroy_cache_entry(), _do_ocsp_validation(), _do_x509_validation(), _dump_cache_data(), _fixup_dnskey_validation(), _get_cache_obj_data(), _get_cert_store(), _load_cache_contents(), _replace_object(), _unlink_object(), _validate_self_signed(), config_free(), tank_count(), tank_delete(), tank_load(), tank_maintain(), tank_size(), tank_start(), tank_stop(), and tank_store().

5.409.2.2 TCHDB* system

Definition at line 16 of file tank.c.

5.409.2.3 TCHDB tanks**

Definition at line 17 of file tank.c.

5.409.2.4 pthread_mutex_t tanks_lock = PTHREAD_MUTEX_INITIALIZER

Definition at line 12 of file tank.c.

Referenced by tank_cycle().

5.409.2.5 uint64_t tanks_next = 0

Definition at line 11 of file tank.c.

Referenced by tank_cycle().

5.409.2.6 uint64_t tanks_num = 4

Definition at line 10 of file tank.c.

Referenced by tank_count(), tank_cycle(), tank_maintain(), tank_size(), tank_start(), tank_stop(), and tank_store().

5.409.2.7 uint8_t tuner

Definition at line 15 of file tank.c.

5.410 src/providers/storage/tokyo.c File Reference

```
#include "magma.h"
```

Functions

- `const char * lib_version_tokyo (void)`
- `bool_t lib_load_tokyo (void)`

Initialize the Tokyo Cabinet library and bind dynamically to the exported functions that are required.

5.410.1 Function Documentation

5.410.1.1 `bool_t lib_load_tokyo (void)`

Initialize the Tokyo Cabinet library and bind dynamically to the exported functions that are required.

Returns:

true on success or false on failure.

Definition at line 19 of file tokyo.c.

References `lib_symbols()`, `M_BIND`, and `symbol_t`.

Referenced by `lib_load()`.

5.410.1.2 `const char* lib_version_tokyo (void)`

Definition at line 11 of file tokyo.c.

References `tcversion_d`.

Referenced by `lib_load()`.

5.411 src/providers/storage/tree.c File Reference

```
#include "magma.h"
```

Functions

- struct `__attribute__((packed))`
- int `tree_cmp` (const char *aptr, int asiz, const char *bptr, int bsiz, void *op)
- uint64_t `tree_count` (void *inx)

Get the number of records in an in-memory tree.

- void * `tree_find` (void *inx, multi_t key)
- bool_t `tree_delete` (void *inx, multi_t key)
- bool_t `tree_insert` (void *inx, multi_t key, void *data)
- bool_t `tree_cursor_next` (tree_cursor_t *cursor)
- void * `tree_cursor_value_next` (tree_cursor_t *cursor)
- void * `tree_cursor_value_active` (tree_cursor_t *cursor)
- multi_t `tree_cursor_key_next` (tree_cursor_t *cursor)
- multi_t `tree_cursor_key_active` (tree_cursor_t *cursor)
- void `tree_cursor_reset` (tree_cursor_t *cursor)
- void `tree_cursor_free` (tree_cursor_t *cursor)
- void * `tree_cursor_alloc` (inx_t *inx)
- void `tree_truncate` (void *inx)
- void `tree_free` (void *inx)
- inx_t * `tree_alloc` (uint64_t options, void *data_free)

Create a new on-memory tree database.

Variables

- tree_cursor_t

5.411.1 Function Documentation

5.411.1.1 struct __attribute__((packed)) [read]

Definition at line 10 of file tree.c.

References inx_t.

5.411.1.2 inx_t* tree_alloc (uint64_t options, void * data_free)

Create a new on-memory tree database.

See also:

tcndbnew2()

Parameters:

options

data_free

Returns:

Definition at line 285 of file tree.c.

References `inx_t`, `log_pedantic`, `mm_alloc()`, `mm_free()`, `tcndbnew2_d`, `tree_cmp()`, `tree_cursor_alloc()`, `tree_cursor_free()`, `tree_cursor_key_active()`, `tree_cursor_key_next()`, `tree_cursor_reset()`, `tree_cursor_value_active()`, `tree_cursor_value_next()`, `tree_delete()`, `tree_find()`, `tree_free()`, `tree_insert()`, and `tree_truncate()`.

Referenced by `inx_alloc()`.

5.411.1.3 int tree_cmp (const char * *aptr*, int *asiz*, const char * *bptr*, int *bsiz*, void * *op*)

The return value is positive if the former is big, negative if the latter is big, 0 if both are equivalent. If asked to search for a NULL key pointer the function will always return 0. This allows us to match every record.

Parameters:

- aptr* a pointer to the region of the first key.
- asize* the length, in bytes, of the region of the first key.
- bptr* a pointer to the region of the second key.
- bsiz* the length, in bytes, of the region of the first key.
- op* an optional pointer to an opaque object.

Returns:

- 0 if the two blocks are identical; otherwise, a signed integer indicating the difference.

Definition at line 28 of file tree.c.

References `cmp_mt_mt()`.

Referenced by `tree_alloc()`.

5.411.1.4 uint64_t tree_count (void * *inx*)

Get the number of records in an in-memory tree. Binary trees.

See also:

- `tcndbrnum()`

Parameters:

- inx*

Returns:

- the record count.

Definition at line 38 of file tree.c.

References `inx_t`, and `tcndbrnum_d`.

5.411.1.5 void* tree_cursor_alloc (inx_t * *inx*)

Definition at line 182 of file tree.c.

References `log_pedantic`, `mm_alloc()`, `tcndbdup_d`, `tcndbiterinit_d`, and `tree_cursor_t`.

Referenced by `tree_alloc()`.

5.411.1.6 void tree_cursor_free (tree_cursor_t * cursor)

Definition at line 169 of file tree.c.

References mm_free(), and tcndbdel_d.

Referenced by tree_alloc().

5.411.1.7 multi_t tree_cursor_key_active (tree_cursor_t * cursor)

Definition at line 151 of file tree.c.

Referenced by tree_alloc().

5.411.1.8 multi_t tree_cursor_key_next (tree_cursor_t * cursor)

Definition at line 144 of file tree.c.

References mt_get_null(), and tree_cursor_next().

Referenced by tree_alloc().

5.411.1.9 bool_t tree_cursor_next (tree_cursor_t * cursor)

Definition at line 109 of file tree.c.

References mm_copy(), mt_get_null(), tcfree_d, tcndbget3_d, and tcndbiternext2_d.

Referenced by tree_cursor_key_next(), and tree_cursor_value_next().

5.411.1.10 void tree_cursor_reset (tree_cursor_t * cursor)

Definition at line 155 of file tree.c.

References tcndbdel_d, tcndbdup_d, and tcndbiterinit_d.

Referenced by tree_alloc().

5.411.1.11 void* tree_cursor_value_active (tree_cursor_t * cursor)

Definition at line 140 of file tree.c.

Referenced by tree_alloc().

5.411.1.12 void* tree_cursor_value_next (tree_cursor_t * cursor)

Definition at line 133 of file tree.c.

References tree_cursor_next().

Referenced by tree_alloc().

5.411.1.13 bool_t tree_delete (void * inx, multi_t key)

Definition at line 53 of file tree.c.

References inx_t, mm_copy(), mt_free(), tcndbgetboth_d, and tcndbout_d.

Referenced by tree_alloc().

5.411.1.14 void* tree_find (void * *inx*, multi_t *key*)

Definition at line 43 of file tree.c.

References *inx_t*, *tcfree_d*, and *tcndbget3_d*.

Referenced by *tree_alloc()*.

5.411.1.15 void tree_free (void * *inx*)

Frees all of the resources used by a tree based index.

Parameters:

inx A pointer to the index that should be freed.

Definition at line 258 of file tree.c.

References *inx_t*, *tcndbdel_d*, and *tree_truncate()*.

Referenced by *tree_alloc()*.

5.411.1.16 bool_t tree_insert (void * *inx*, multi_t *key*, void * *data*)

Makes a copy of the key and then attempts to add the new entry to the tree index. Note that because this is a tree index, duplicates are not allowed, so if the key already exists, false is returned.

Parameters:

inx The index were adding the entry too.

key The retrieval key expressed as a [multi_t](#).

data The data buffer being stored.

Returns:

Returns true if the entry was added, or false to indicate an existing duplicate key or an error.

Definition at line 87 of file tree.c.

References *inx_t*, *log_info*, *mt_dupe()*, *mt_free()*, *mt_is_empty()*, and *tcndbputkeep_d*.

Referenced by *tree_alloc()*.

5.411.1.17 void tree_truncate (void * *inx*)

Truncates a tree based index.

Parameters:

inx A pointer to the index that should be truncated.

Definition at line 208 of file tree.c.

References *count*, *inx_t*, *length*, *mt_free()*, *tclistdel_d*, *tclistnum_d*, *tclistval_d*, *tctreeclear_d*, *tctreekeys_d*, and *tctreevals_d*.

Referenced by *tree_alloc()*, and *tree_free()*.

5.411.2 Variable Documentation

5.411.2.1 `tree_cursor_t`

Definition at line 15 of file `tree.c`.

Referenced by `tree_cursor_alloc()`.

5.412 src/providers/symbols.c File Reference

```
#include "magma.h"
```

Functions

- [STACK_OF](#) (SSL_COMP)
- void [lib_unload](#) (void)
Unload magmad.so from memory.
- bool_t [lib_load](#) (void)
Load magmad.so dynamically and resolve all external dependencies from 3rd party providers.

Variables

- memcached_return_t(* [memcached_flush_d](#))(memcached_st *ptr, time_t expiration) = NULL
MEMCACHED.
- void(* [memcached_free_d](#))(memcached_st *ptr) = NULL
- const char *(* [memcached_lib_version_d](#))(void) = NULL
- memcached_st *(* [memcached_create_d](#))(memcached_st *ptr) = NULL
- const char *(* [memcached_strerror_d](#))(const memcached_st *ptr, memcached_return_t rc) = NULL
- memcached_return_t(* [memcached_behavior_set_d](#))(memcached_st *ptr, const memcached_behavior_t flag, uint64_t data) = NULL
- memcached_return_t(* [memcached_delete_d](#))(memcached_st *ptr, const char *key, size_t key_length, time_t expiration) = NULL
- memcached_return_t(* [memcached_server_add_with_weight_d](#))(memcached_st *ptr, const char *hostname, in_port_t port, uint32_t weight) = NULL
- memcached_return_t(* [memcached_decrement_d](#))(memcached_st *ptr, const char *key, size_t key_length, uint32_t offset, uint64_t *value) = NULL
- memcached_return_t(* [memcached_increment_d](#))(memcached_st *ptr, const char *key, size_t key_length, uint32_t offset, uint64_t *value) = NULL
- char *(* [memcached_get_d](#))(memcached_st *ptr, const char *key, size_t key_length, size_t *value_length, uint32_t *flags, memcached_return_t *error) = NULL
- memcached_return_t(* [memcached_add_d](#))(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL
- memcached_return_t(* [memcached_set_d](#))(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL
- memcached_return_t(* [memcached_append_d](#))(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL
- memcached_return_t(* [memcached_prepend_d](#))(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL
- memcached_return_t(* [memcached_replace_d](#))(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL
- memcached_return_t(* [memcached_cas_d](#))(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags, uint64_t cas) = NULL
- memcached_return_t(* [memcached_decrement_with_initial_d](#))(memcached_st *ptr, const char *key, size_t key_length, uint64_t offset, uint64_t initial, time_t expiration, uint64_t *value) = NULL
- memcached_return_t(* [memcached_increment_with_initial_d](#))(memcached_st *ptr, const char *key, size_t key_length, uint64_t offset, uint64_t initial, time_t expiration, uint64_t *value) = NULL
- const char *(* [BZ2_bzlibVersion_d](#))(void) = NULL
BZIP.

- `int(* BZ2_bzBuffToBuffDecompress_d)(char *dest, unsigned int *destLen, char *source, unsigned int sourceLen, int small, int verbosity) = NULL`
- `int(* BZ2_bzBuffToBuffCompress_d)(char *dest, unsigned int *destLen, char *source, unsigned int sourceLen, int blockSize100k, int verbosity, int workFactor) = NULL`
- `void(* cl_shutdown_d)(void) = NULL`

CLAMAV.

- `int(* lt_dlexit_d)(void) = NULL`
- `const char *(* cl_retver_d)(void) = NULL`
- `int(* cl_init_d)(unsigned int initoptions) = NULL`
- `const char *(* cl_strerror_d)(int clerror) = NULL`
- `struct cl_engine *(* cl_engine_new_d)(void) = NULL`
- `int(* cl_statfree_d)(struct cl_stat *dbstat) = NULL`
- `int(* cl_engine_free_d)(struct cl_engine *engine) = NULL`
- `int(* cl_engine_compile_d)(struct cl_engine *engine) = NULL`
- `int(* cl_statchkdir_d)(const struct cl_stat *dbstat) = NULL`
- `int(* cl_statinidir_d)(const char *dirname, struct cl_stat *dbstat) = NULL`
- `int(* cl_countsigs_d)(const char *path, unsigned int countoptions, unsigned int *sigs) = NULL`
- `int(* cl_engine_set_num_d)(struct cl_engine *engine, enum cl_engine_field field, long long num) = NULL`
- `int(* cl_engine_set_str_d)(struct cl_engine *engine, enum cl_engine_field field, const char *str) = NULL`
- `int(* cl_load_d)(const char *path, struct cl_engine *engine, unsigned int *signo, unsigned int doptions) = NULL`
- `int(* cl_scandesc_d)(int desc, const char **virname, unsigned long int *scanned, const struct cl_engine *engine, unsigned int scanoptions) = NULL`
- `const char *(* dspam_version_d)(void) = NULL`

DSPAM.

- `int(* dspam_detach_d)(DSPAM_CTX *CTX) = NULL`
- `void(* dspam_destroy_d)(DSPAM_CTX *CTX) = NULL`
- `int(* dspam_init_driver_d)(DRIVER_CTX *DTX) = NULL`
- `int(* dspam_shutdown_driver_d)(DRIVER_CTX *DTX) = NULL`
- `int(* dspam_attach_d)(DSPAM_CTX *CTX, void *dbh) = NULL`
- `int(* dspam_process_d)(DSPAM_CTX *CTX, const char *message) = NULL`
- `DSPAM_CTX *(* dspam_create_d)(const char *username, const char *group, const char *home, int operating_mode, u_int32_t flags) = NULL`
- `DKIM_STAT(* dkim_eoh_d)(DKIM *dkim) = NULL`
- `void(* dkim_close_d)(DKIM_LIB *lib) = NULL`
- `uint32_t(* dkim_libversion_d)(void) = NULL`
- `DKIM_STAT(* dkim_free_d)(DKIM *dkim) = NULL`
- `char *(* dkim_geterror_d)(DKIM *dkim) = NULL`
- `DKIM_STAT(* dkim_eom_d)(DKIM *dkim, _Bool *testkey) = NULL`
- `const char *(* dkim_getresultstr_d)(DKIM_STAT result) = NULL`
- `DKIM_STAT(* dkim_body_d)(DKIM *dkim, u_char *buf, size_t len) = NULL`
- `DKIM_STAT(* dkim_header_d)(DKIM *dkim, u_char *hdr, size_t len) = NULL`
- `void(* dkim_mfree_d)(DKIM_LIB *libhandle, void *closure, void *ptr) = NULL`
- `DKIM_STAT(* dkim_chunk_d)(DKIM *dkim, unsigned char *chunkp, size_t len) = NULL`
- `DKIM_STAT(* dkim_getsighdrx_d)(DKIM *dkim, u_char *buf, size_t len, size_t initial) = NULL`
- `int(* dkim_test_dns_put_d)(DKIM *dkim, int class, int type, int prec, u_char *name, u_char *data) = NULL`
- `DKIM *(* dkim_verify_d)(DKIM_LIB *libhandle, const unsigned char *id, void *memclosure, DKIM_STAT *statp) = NULL`
- `DKIM_LIB *(* dkim_init_d)(void *(*mallocf)(void *closure, size_t nbytes), void(*freef)(void *closure, void *p)) = NULL`
- `DKIM *(* dkim_sign_d)(DKIM_LIB *libhandle, const unsigned char *id, void *memclosure, const dkim_sigkey_t secretkey, const unsigned char *selector, const unsigned char *domain, dkim_canon_t hdr_canon_alg, dkim_canon_t body_canon_alg, dkim_alg_t sign_alg, off_t length, DKIM_STAT *statp) = NULL`
- `FT_Error(* FT_Done_FreeType_d)(FT_Library library) = NULL`

FreeType.

- FT_Error(* FT_Init_FreeType_d)(FT_Library *alibrary) = NULL
- void(* FT_Library_Version_d)(FT_Library library, FT_Int *amajor, FT_Int *aminor, FT_Int *apatch) = NULL
- const char *(* gdVersionString_d)(void) = NULL

GD.

- void(* gdFree_d)(void *m) = NULL
- void *(* gdImageGifPtr_d)(gdImagePtr im, int *size) = NULL
- void(* gdImageDestroy_d)(gdImagePtr im) = NULL
- void *(* gdImageJpegPtr_d)(gdImagePtr im, int *size, int quality) = NULL
- void(* gdImageSetPixel_d)(gdImagePtr im, int x, int y, int color) = NULL
- gdImagePtr(* gdImageCreate_d)(int sx, int sy) = NULL
- int(* gdImageColorResolve_d)(gdImagePtr im, int r, int g, int b) = NULL
- char *(* gdImageStringFT_d)(gdImage *im, int *breect, int fg, char *fontlist, double psize, double angle, int x, int y, char *string) = NULL
- const char *(* jpeg_version_d)(void) = NULL

JPEG.

- const char *(* lzo_version_string_d)(void) = NULL

LZO.

- int(* __lzo_init_v2_d)(unsigned, int, int, int, int, int, int, int, int) = NULL
- lzo_uint32(* lzo_adler32_d)(lzo_uint32_adler, const lzo_bytewp_buf, lzo_uint _len) = NULL
- int(* lzo1x_1_compress_d)(const lzo_byte *src, lzo_uint src_len, lzo_byte *dst, lzo_uintp dst_len, lzo_voidp wrkmem) = NULL
- int(* lzo1x_decompress_safe_d)(const lzo_byte *src, lzo_uint src_len, lzo_byte *dst, lzo_uintp dst_len, lzo_voidp wrkmem) = NULL
- void(* my_once_free_d)(void) = NULL

MYSQL.

- void(* mysql_server_end_d)(void) = NULL
- void(* mysql_thread_end_d)(void) = NULL
- int(* mysql_ping_d)(MYSQL *mysql) = NULL
- my_bool(* mysql_embedded_d)(void) = NULL
- void(* mysql_close_d)(MYSQL *mysql) = NULL
- my_bool(* mysql_thread_init_d)(void) = NULL
- MYSQL *(* mysql_init_d)(MYSQL *mysql) = NULL
- unsigned int(* mysql_thread_safe_d)(void) = NULL
- int(* mysql_stmt_fetch_d)(MYSQL_STMT *stmt) = NULL
- my_bool(* mysql_stmt_close_d)(MYSQL_STMT *) = NULL
- unsigned int(* mysql_errno_d)(MYSQL *mysql) = NULL
- const char *(* mysql_error_d)(MYSQL *mysql) = NULL
- int(* mysql_stmt_execute_d)(MYSQL_STMT *stmt) = NULL
- void(* mysql_free_result_d)(MYSQL_RES *result) = NULL
- my_bool(* mysql_stmt_reset_d)(MYSQL_STMT *stmt) = NULL
- my_ulonglong(* mysql_insert_id_d)(MYSQL *mysql) = NULL
- unsigned long(* mysql_thread_id_d)(MYSQL *mysql) = NULL
- MYSQL_STMT *(* mysql_stmt_init_d)(MYSQL *mysql) = NULL
- MYSQL_ROW(* mysql_fetch_row_d)(MYSQL_RES *result) = NULL
- unsigned long(* mysql_get_client_version_d)(void) = NULL
- MYSQL_RES *(* mysql_store_result_d)(MYSQL *mysql) = NULL
- int(* mysql_stmt_store_result_d)(MYSQL_STMT *stmt) = NULL
- my_ulonglong(* mysql_affected_rows_d)(MYSQL *mysql) = NULL
- my_ulonglong(* mysql_num_rows_d)(MYSQL_RES *result) = NULL

- unsigned int(* [mysql_stmt_errno_d](#))(MYSQL_STMT *stmt) = NULL
- const char *([mysql_get_server_info_d](#))(MYSQL *mysql) = NULL
- const char *([mysql_stmt_error_d](#))(MYSQL_STMT *stmt) = NULL
- my_bool(* [mysql_stmt_free_result_d](#))(MYSQL_STMT *stmt) = NULL
- unsigned int(* [mysql_num_fields_d](#))(MYSQL_RES *result) = NULL
- my_ulonglong(* [mysql_stmt_num_rows_d](#))(MYSQL_STMT *stmt) = NULL
- MYSQL_FIELD *([mysql_fetch_field_d](#))(MYSQL_RES *result) = NULL
- const char *([mysql_character_set_name_d](#))(MYSQL *mysql) = NULL
- my_ulonglong(* [mysql_stmt_insert_id_d](#))(MYSQL_STMT *stmt) = NULL
- my_ulonglong(* [mysql_stmt_affected_rows_d](#))(MYSQL_STMT *stmt) = NULL
- MYSQL_RES *([mysql_stmt_result_metadata_d](#))(MYSQL_STMT *stmt) = NULL
- int(* [mysql_server_init_d](#))(int argc, char **argv, char **groups) = NULL
- int(* [mysql_set_character_set_d](#))(MYSQL *mysql, const char *csname) = NULL
- my_bool(* [mysql_stmt_bind_param_d](#))(MYSQL_STMT *stmt, MYSQL_BIND *bind) = NULL
- my_bool(* [mysql_stmt_bind_result_d](#))(MYSQL_STMT *stmt, MYSQL_BIND *bind) = NULL
- int(* [mysql_options_d](#))(MYSQL *mysql, enum mysql_option option, const void *arg) = NULL
- int(* [mysql_real_query_d](#))(MYSQL *mysql, const char *query, unsigned long [length](#)) = NULL
- int(* [mysql_stmt_prepare_d](#))(MYSQL_STMT *stmt, const char *query, unsigned long [length](#)) = NULL
- unsigned long(* [mysql_escape_string_d](#))(char *to, const char *from, unsigned long [length](#)) = NULL
- my_bool(* [mysql_stmt_attr_set_d](#))(MYSQL_STMT *stmt, enum enum_stmt_attr_type attr_type, const void *attr) = NULL
- MYSQL *([mysql_real_connect_d](#))(MYSQL *mysql, const char *name, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag) = NULL
- DH *([DH_new_d](#))(void) = NULL

OPENSSL

- char ** [SSL_version_str_d](#) = NULL
- RSA *([RSA_new_d](#))(void) = NULL
- void(* [DH_free_d](#))(DH *dh) = NULL
- int(* [BIO_free_d](#))(BIO *a) = NULL
- int(* [RAND_status_d](#))(void) = NULL
- void(* [RSA_free_d](#))(RSA *r) = NULL
- void(* [EVP_cleanup_d](#))(void) = NULL
- void(* [OBJ_cleanup_d](#))(void) = NULL
- void(* [BN_free_d](#))(BIGNUM *a) = NULL
- void(* [RAND_cleanup_d](#))(void) = NULL
- void(* [SSL_free_d](#))(SSL *ssl) = NULL
- int(* [SSL_accept_d](#))(SSL *ssl) = NULL
- void *([sk_pop_d](#))(_STACK *st) = NULL
- BN_CTX *([BN_CTX_new_d](#))(void) = NULL
- int(* [SSL_connect_d](#))(SSL *ssl) = NULL
- EC_KEY *([EC_KEY_new_d](#))(void) = NULL
- void(* [CRYPTO_free_d](#))(void *) = NULL
- void(* [ENGINE_cleanup_d](#))(void) = NULL
- int(* [SHA1_Init_d](#))(SHA_CTX *c) = NULL
- void(* [BIO_free_all_d](#))(BIO *a) = NULL
- int(* [CRYPTO_num_locks_d](#))(void) = NULL
- int(* [SSL_library_init_d](#))(void) = NULL
- int(* [SSL_want_d](#))(const SSL *s) = NULL
- int(* [SSL_shutdown_d](#))(SSL *ssl) = NULL
- void(* [ERR_clear_error_d](#))(void) = NULL
- int(* [sk_num_d](#))(const _STACK *) = NULL
- void(* [BIO_sock_cleanup_d](#))(void) = NULL
- void(* [ERR_free_strings_d](#))(void) = NULL

- `SSL *(* SSL_new_d)(SSL_CTX *ctx) = NULL`
- `const EVP_MD *(* EVP_md4_d)(void) = NULL`
- `const EVP_MD *(* EVP_md5_d)(void) = NULL`
- `const EVP_MD *(* EVP_sha_d)(void) = NULL`
- `void(* COMP_zlib_cleanup_d)(void) = NULL`
- `int(* SSL_get_fd_d)(const SSL *s) = NULL`
- `int(* SSL_do_handshake_d)(SSL *s) = NULL`
- `void(* BN_CTX_free_d)(BN_CTX *ctx) = NULL`
- `int(* SSL_get_rfd_d)(const SSL *s) = NULL`
- `const EVP_MD *(* EVP_sha1_d)(void) = NULL`
- `void(* EC_KEY_free_d)(EC_KEY *key) = NULL`
- `EVP_PKEY *(* EVP_PKEY_new_d)(void) = NULL`
- `void(* BN_CTX_start_d)(BN_CTX *ctx) = NULL`
- `const char *(* OBJ_nid2sn_d)(int n) = NULL`
- `int(* SHA256_Init_d)(SHA256_CTX *c) = NULL`
- `int(* SHA512_Init_d)(SHA512_CTX *c) = NULL`
- `int(* SSL_set_fd_d)(SSL *s, int fd) = NULL`
- `const EVP_MD *(* EVP_sha224_d)(void) = NULL`
- `const EVP_MD *(* EVP_sha256_d)(void) = NULL`
- `const EVP_MD *(* EVP_sha384_d)(void) = NULL`
- `const EVP_MD *(* EVP_sha512_d)(void) = NULL`
- `void(* OBJ_NAME_cleanup_d)(int type) = NULL`
- `void(* SSL_CTX_free_d)(SSL_CTX *ctx) = NULL`
- `int(* SSL_pending_d)(const SSL *ssl) = NULL`
- `int(* BN_num_bits_d)(const BIGNUM *) = NULL`
- `int(* X509_get_ext_count_d)(X509 *x) = NULL`
- `RSA *(* RSAPublicKey_dup_d)(RSA *rsa) = NULL`
- `char *(* BN_bn2dec_d)(const BIGNUM *a) = NULL`
- `char *(* BN_bn2hex_d)(const BIGNUM *a) = NULL`
- `int(* EVP_MD_size_d)(const EVP_MD *md) = NULL`
- `unsigned long(* ERR_get_error_d)(void) = NULL`
- `void(* CONF_modules_unload_d)(int all) = NULL`
- `void(* HMAC_CTX_init_d)(HMAC_CTX *ctx) = NULL`
- `void(* SSL_load_error_strings_d)(void) = NULL`
- `int(* EVP_MD_type_d)(const EVP_MD *md) = NULL`
- `void(* ECDSA_SIG_free_d)(ECDSA_SIG *a) = NULL`
- `X509_STORE *(* X509_STORE_new_d)(void) = NULL`
- `void(* SSL_set_accept_state_d)(SSL *s) = NULL`
- `void(* SSL_set_connect_state_d)(SSL *s) = NULL`
- `unsigned long(* ERR_peek_error_d)(void) = NULL`
- `const EVP_MD *(* EVP_ripemd160_d)(void) = NULL`
- `const char *(* SSLeay_version_d)(int t) = NULL`
- `void(* ERR_load_crypto_strings_d)(void) = NULL`
- `void(* ERR_print_errors_fp_d)(FILE *fp) = NULL`
- `BIO *(* SSL_get_wbio_d)(const SSL *ssl) = NULL`
- `void(* EC_GROUP_free_d)(EC_GROUP *group) = NULL`
- `void(* EC_POINT_free_d)(EC_POINT *point) = NULL`
- `void(* X509_STORE_free_d)(X509_STORE *v) = NULL`
- `int(* SSL_get_read_ahead_d)(const SSL *s) = NULL`
- `int(* DH_check_d)(const DH *dh, int *ret) = NULL`
- `int(* EC_KEY_generate_key_d)(EC_KEY *key) = NULL`
- `void(* ASN1_STRING_TABLE_cleanup_d)(void) = NULL`
- `void(* HMAC_CTX_cleanup_d)(HMAC_CTX *ctx) = NULL`

- `int(* SSL_get_shutdown_d)(const SSL *ssl) = NULL`
- `void (* sk_value_d)(const _STACK *, int) = NULL`
- `void(* CRYPTO_cleanup_all_ex_data_d)(void) = NULL`
- `void(* EVP_MD_CTX_init_d)(EVP_MD_CTX *ctx) = NULL`
- `OCSP_REQUEST *(* OCSP_REQUEST_new_d)(void) = NULL`
- `int(* EC_KEY_check_key_d)(const EC_KEY *key) = NULL`
- `int(* EVP_MD_CTX_cleanup_d)(EVP_MD_CTX *ctx) = NULL`
- `void(* OCSP_REQUEST_free_d)(OCSP_REQUEST *a) = NULL`
- `const EVP_CIPHER *(* EVP_aes_256_gcm_d)(void) = NULL`
- `int(* SSL_peek_d)(SSL *ssl, void *buf, int num) = NULL`
- `const EVP_CIPHER *(* EVP_aes_256_cbc_d)(void) = NULL`
- `void(* SSL_set_read_ahead_d)(SSL *s, int yes) = NULL`
- `EVP_CIPHER_CTX *(* EVP_CIPHER_CTX_new_d)(void) = NULL`
- `int(* OCSP_check_nonce_d)(void *req, void *bs) = NULL`
- `int(* X509_verify_cert_d)(X509_STORE_CTX *ctx) = NULL`
- `void(* EC_GROUP_clear_free_d)(EC_GROUP *group) = NULL`
- `void(* OCSP_RESPONSE_free_d)(OCSP_RESPONSE *a) = NULL`
- `X509_STORE_CTX *(* X509_STORE_CTX_new_d)(void) = NULL`
- `X509_NAME *(* X509_get_subject_name_d)(X509 *a) = NULL`
- `EC_KEY *(* EC_KEY_new_by_curve_name_d)(int nid) = NULL`
- `int(* BN_hex2bn_d)(BIGNUM **a, const char *str) = NULL`
- `int(* SSL_read_d)(SSL *ssl, void *buf, int num) = NULL`
- `int(* i2d_X509_d)(X509 *a, unsigned char **out) = NULL`
- `const char *(* SSL_get_version_d)(const SSL *s) = NULL`
- `int(* RAND_bytes_d)(unsigned char *buf, int num) = NULL`
- `void(* EVP_CIPHER_CTX_init_d)(EVP_CIPHER_CTX *a) = NULL`
- `void(* OCSP_BASICRESP_free_d)(OCSP_BASICRESP *a) = NULL`
- `void(* EVP_CIPHER_CTX_free_d)(EVP_CIPHER_CTX *a) = NULL`
- `X509_LOOKUP_METHOD *(* X509_LOOKUP_file_d)(void) = NULL`
- `int(* BN_cmp_d)(const BIGNUM *a, const BIGNUM *b) = NULL`
- `const SSL_METHOD *(* TLSv1_server_method_d)(void) = NULL`
- `int(* EVP_CIPHER_nid_d)(const EVP_CIPHER *cipher) = NULL`
- `void(* OPENSSL_add_all_algorithms_noconf_d)(void) = NULL`
- `int(* SSL_get_error_d)(const SSL *s, int ret_code) = NULL`
- `const SSL_METHOD *(* SSLv23_client_method_d)(void) = NULL`
- `const SSL_METHOD *(* SSLv23_server_method_d)(void) = NULL`
- `X509 *(* SSL_get_peer_certificate_d)(const SSL *s) = NULL`
- `int(* EVP_CIPHER_CTX_cleanup_d)(EVP_CIPHER_CTX *a) = NULL`
- `const char *(* OCSP_response_status_str_d)(long s) = NULL`
- `int(* OCSP_response_status_d)(OCSP_RESPONSE *resp) = NULL`
- `int(* SHA1_Final_d)(unsigned char *md, SHA_CTX *c) = NULL`
- `void(* X509_STORE_CTX_free_d)(X509_STORE_CTX *ctx) = NULL`
- `BIO *(* BIO_new_socket_d)(int sock, int close_flag) = NULL`
- `EC_GROUP *(* EC_GROUP_new_by_curve_name_d)(int nid) = NULL`
- `EC_POINT *(* EC_POINT_new_d)(const EC_GROUP *group) = NULL`
- `int(* BN_bn2bin_d)(const BIGNUM *, unsigned char *) = NULL`
- `BIO *(* BIO_new_fp_d)(FILE *stream, int close_flag) = NULL`
- `X509_EXTENSION *(* X509_get_ext_d)(X509 *x, int loc) = NULL`
- `SSL_CTX *(* SSL_CTX_new_d)(const SSL_METHOD *method) = NULL`
- `void(* SSL_set_bio_d)(SSL *ssl, BIO *rbio, BIO *wbio) = NULL`
- `unsigned char *(* ASN1_STRING_data_d)(ASN1_STRING *x) = NULL`
- `int(* BN_bn2mpi_d)(const BIGNUM *a, unsigned char *to) = NULL`
- `int(* SSL_CTX_check_private_key_d)(const SSL_CTX *ctx) = NULL`

- `int(* SSL_write_d)(SSL *ssl, const void *buf, int num) = NULL`
- `void(* sk_pop_free_d)(_STACK *st, void(*func)(void *)) = NULL`
- `int(* X509_STORE_CTX_get_error_d)(X509_STORE_CTX *ctx) = NULL`
- `void(* OCSRP_request_add0_id_d)(void *req, void *cid) = NULL`
- `int(* EVP_CIPHER_iv_length_d)(const EVP_CIPHER *cipher) = NULL`
- `const char *(* X509_verify_cert_error_string_d)(long n) = NULL`
- `int(* SHA256_Final_d)(unsigned char *md, SHA256_CTX *c) = NULL`
- `int(* SHA512_Final_d)(unsigned char *md, SHA512_CTX *c) = NULL`
- `int(* X509_check_issued_d)(X509 *issuer, X509 *subject) = NULL`
- `SSL_CIPHER *(* SSL_get_current_cipher_d)(const SSL *ssl) = NULL`
- `int(* EVP_CIPHER_block_size_d)(const EVP_CIPHER *cipher) = NULL`
- `int(* EVP_CIPHER_key_length_d)(const EVP_CIPHER *cipher) = NULL`
- `void(* OCSRP_response_get1_basic_d)(OCSRP_RESPONSE *resp) = NULL`
- `const EC_GROUP *(* EC_KEY_get0_group_d)(const EC_KEY *key) = NULL`
- `const EVP_MD *(* EVP_get_digestbyname_d)(const char *name) = NULL`
- `long(* SSL_ctrl_d)(SSL *s, int cmd, long larg, void *parg) = NULL`
- `int(* i2o_ECPublicKey_d)(EC_KEY *key, unsigned char **out) = NULL`
- `int(* SSL_CTX_set_cipher_list_d)(SSL_CTX *, const char *str) = NULL`
- `int(* i2d_ECPrivateKey_d)(EC_KEY *key, unsigned char **out) = NULL`
- `char *(* SSL_CIPHER_get_version_d)(const SSL_CIPHER *cipher) = NULL`
- `int(* EVP_CIPHER_CTX_iv_length_d)(const EVP_CIPHER_CTX *ctx) = NULL`
- `int(* EVP_DigestInit_d)(EVP_MD_CTX *ctx, const EVP_MD *type) = NULL`
- `int(* X509_STORE_CTX_get_error_depth_d)(X509_STORE_CTX *ctx) = NULL`
- `int(* EC_KEY_set_group_d)(EC_KEY *key, const EC_GROUP *group) = NULL`
- `int(* EVP_CIPHER_CTX_block_size_d)(const EVP_CIPHER_CTX *ctx) = NULL`
- `int(* EVP_CIPHER_CTX_key_length_d)(const EVP_CIPHER_CTX *ctx) = NULL`
- `int(* RAND_load_file_d)(const char *filename, long max_bytes) = NULL`
- `void(* ERR_remove_thread_state_d)(const CRYPTO_THREADID *tid) = NULL`
- `unsigned long(* EVP_CIPHER_flags_d)(const EVP_CIPHER *cipher) = NULL`
- `int(* i2d_OCSP_CERTID_d)(OCSP_CERTID *a, unsigned char **out) = NULL`
- `int(* OCSRP_REQ_CTX_set1_req_d)(OCSRP_REQ_CTX *rctx, void *req) = NULL`
- `struct stack_st_OPENSSL_STRING *(* X509_get1_ocsp_d)(X509 *x) = NULL`
- `void(* X509_email_free_d)(struct stack_st_OPENSSL_STRING *sk) = NULL`
- `const BIGNUM *(* EC_KEY_get0_private_key_d)(const EC_KEY *key) = NULL`
- `const EVP_CIPHER *(* EVP_get_cipherbyname_d)(const char *name) = NULL`
- `int(* EVP_PKEY_set1_RSA_d)(EVP_PKEY *pkey, struct rsa_st *key) = NULL`
- `int(* SHA1_Update_d)(SHA_CTX *c, const void *data, size_t len) = NULL`
- `const char *(* SSL_CIPHER_get_name_d)(const SSL_CIPHER *cipher) = NULL`
- `BIGNUM *(* BN_mpi2bn_d)(unsigned char *s, int len, BIGNUM *ret) = NULL`
- `const EC_POINT *(* EC_KEY_get0_public_key_d)(const EC_KEY *key) = NULL`
- `int(* EC_GROUP_precompute_mult_d)(EC_GROUP *group, BN_CTX *ctx) = NULL`
- `int(* EC_KEY_set_private_key_d)(EC_KEY *key, const BIGNUM *prv) = NULL`
- `int(* EVP_CIPHER_CTX_set_padding_d)(EVP_CIPHER_CTX *c, int pad) = NULL`
- `long(* SSL_CTX_callback_ctrl_d)(SSL_CTX *, int, void(*)(void)) = NULL`
- `char *(* X509_NAME_online_d)(X509_NAME *a, char *buf, int len) = NULL`
- `size_t(* BUF_strlcat_d)(char *dst, const char *src, size_t siz) = NULL`

5.412.1 Function Documentation

5.412.1.1 `bool_t lib_load (void)`

Load magmad.so dynamically and resolve all external dependencies from 3rd party providers.

Returns:

false on failure or true on success.

Definition at line 658 of file symbols.c.

References `build_commit()`, `build_stamp()`, `build_version()`, `magma_t::config`, `CONSTANT`, `magma_t::file`, `host_platform()`, `host_version()`, `lib_load_bzip()`, `lib_load_cache()`, `lib_load_clamav()`, `lib_load_dkim()`, `lib_load_dspam()`, `lib_load_freetype()`, `lib_load_gd()`, `lib_load_jansson()`, `lib_load_jpeg()`, `lib_load_lzo()`, `lib_load_mysql()`, `lib_load_openssl()`, `lib_load_png()`, `lib_load_spf()`, `lib_load_tokyo()`, `lib_load_utf8proc()`, `lib_load_xml()`, `lib_load_zlib()`, `lib_magma`, `lib_version_bzip()`, `lib_version_cache()`, `lib_version_clamav()`, `lib_version_dkim()`, `lib_version_dspam()`, `lib_version_freetype()`, `lib_version_gd()`, `lib_version_jansson()`, `lib_version_jpeg()`, `lib_version_lzo()`, `lib_version_mysql()`, `lib_version_openssl()`, `lib_version_png()`, `lib_version_spf()`, `lib_version_tokyo()`, `lib_version_utf8proc()`, `lib_version_xml()`, `lib_version_zlib()`, `magma_t::library`, `log_critical`, `log_options`, `M_LOG_FILE_DISABLE`, `M_LOG_FUNCTION_DISABLE`, `M_LOG_INFO`, `M_LOG_LINE_DISABLE`, `M_LOG_STACK_TRACE_DISABLE`, `M_LOG_TIME_DISABLE`, `magma`, `MANAGEDBUF`, `ns_empty()`, `NULLER`, `serv_charset_mysql()`, `serv_schema_mysql()`, `serv_type_mysql()`, `serv_version_mysql()`, `st_char_get()`, and `st_cmp_ci_eq()`.

Referenced by `process_start()`.

5.412.1.2 `void lib_unload (void)`

Unload magmad.so from memory.

Returns:

This function returns no value.

Definition at line 645 of file symbols.c.

References `lib_magma`, `magma_t::library`, `magma`, and `magma_t::unload`.

Referenced by `process_stop()`.

5.412.1.3 `STACK_OF (SSL_COMP)`

Definition at line 339 of file symbols.c.

References `lib_magma`, `log_critical`, `NULLER`, `st_cmp_cs_eq()`, `tcversion_d`, and `xmlParserVersion_d`.

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.412.2 Variable Documentation

5.412.2.1 `int(* __lzo_init_v2_d)(unsigned, int, int, int, int, int, int, int, int, int) = NULL`

Referenced by `lib_load_lzo()`.

5.412.2.2 `unsigned char*(* ASN1_STRING_data_d)(ASN1_STRING *x) = NULL`

Referenced by `_get_cert_subject_cn()`.

5.412.2.3 void(* ASN1_STRING_TABLE_cleanup_d)(void) = NULL

Referenced by ssl_stop().

5.412.2.4 void(* BIO_free_all_d)(BIO *a) = NULL

Referenced by _do_ocsp_validation().

5.412.2.5 int(* BIO_free_d)(BIO *a) = NULL

Referenced by _do_ocsp_validation(), and _dump_ocsp_response_cb().

5.412.2.6 BIO*(* BIO_new_fp_d)(FILE *stream, int close_flag) = NULL

Referenced by _do_ocsp_validation(), and _dump_ocsp_response_cb().

5.412.2.7 BIO*(* BIO_new_socket_d)(int sock, int close_flag) = NULL

Referenced by _do_ocsp_validation(), tls_client_alloc(), and tls_server_alloc().

5.412.2.8 void(* BIO_sock_cleanup_d)(void) = NULL

Referenced by ssl_stop().

5.412.2.9 int(* BN_bn2bin_d)(const BIGNUM *, unsigned char *) = NULL

Referenced by _encode_rsa_pubkey(), deprecated_ecies_key_private_bin(), ecies_key_private_bin(), and secp256k1_private_get().

5.412.2.10 char*(* BN_bn2dec_d)(const BIGNUM *a) = NULL

5.412.2.11 char*(* BN_bn2hex_d)(const BIGNUM *a) = NULL

Referenced by _dump_ocsp_response_cb(), deprecated_ecies_key_private_hex(), and ecies_key_private_hex().

5.412.2.12 int(* BN_bn2mpi_d)(const BIGNUM *a, unsigned char *to) = NULL

5.412.2.13 int(* BN_cmp_d)(const BIGNUM *a, const BIGNUM *b) = NULL

5.412.2.14 void(* BN_CTX_free_d)(BN_CTX *ctx) = NULL

Referenced by secp256k1_private_set(), and secp256k1_public_set().

5.412.2.15 BN_CTX*(* BN_CTX_new_d)(void) = NULL

Referenced by secp256k1_private_set(), and secp256k1_public_set().

5.412.2.16 void(* BN_CTX_start_d)(BN_CTX *ctx) = NULL

Referenced by secp256k1_private_set(), and secp256k1_public_set().

5.412.2.17 void(* BN_free_d)(BIGNUM *a) = NULL

Referenced by _decode_rsa_pubkey(), _dump_ocsp_response_cb(), _get_rsa_dnskey(), deprecated_ecies_key_private(), ecies_key_private(), and secp256k1_private_set().

5.412.2.18 int(* BN_hex2bn_d)(BIGNUM **a, const char *str) = NULL

Referenced by deprecated_ecies_key_private(), and ecies_key_private().

5.412.2.19 BIGNUM*(* BN_mpi2bn_d)(unsigned char *s, int len, BIGNUM *ret) = NULL

5.412.2.20 int(* BN_num_bits_d)(const BIGNUM *) = NULL

Referenced by _encode_rsa_pubkey(), _get_rsa_dnskey(), and secp256k1_private_get().

5.412.2.21 size_t(* BUF_strlcat_d)(char *dst, const char *src, size_t siz) = NULL

Referenced by _push_error_stack_openssl().

5.412.2.22 int(* BZ2_bzBuffToBuffCompress_d)(char *dest, unsigned int *destLen, char *source, unsigned int sourceLen, int blockSize100k, int verbosity, int workFactor) = NULL

Referenced by compress_bzip().

5.412.2.23 int(* BZ2_bzBuffToBuffDecompress_d)(char *dest, unsigned int *destLen, char *source, unsigned int sourceLen, int small, int verbosity) = NULL

Referenced by decompress_bzip().

5.412.2.24 const char*(* BZ2_bzlibVersion_d)(void) = NULL

BZIP.

Referenced by lib_load_bzip().

5.412.2.25 int(* cl_countsigs_d)(const char *path, unsigned int countoptions, unsigned int *sigs) = NULL

Referenced by virus_sigs_total().

5.412.2.26 int(* cl_engine_compile_d)(struct cl_engine *engine) = NULL

Referenced by virus_engine_create().

5.412.2.27 int(* cl_engine_free_d)(struct cl_engine *engine) = NULL

Referenced by virus_engine_create(), and virus_engine_destroy().

5.412.2.28 struct cl_engine*(* cl_engine_new_d)(void) = NULL

Referenced by virus_engine_create().

5.412.2.29 `int(* cl_engine_set_num_d)(struct cl_engine *engine, enum cl_engine_field field, long long num) = NULL`

Referenced by `virus_engine_create()`.

5.412.2.30 `int(* cl_engine_set_str_d)(struct cl_engine *engine, enum cl_engine_field field, const char *str) = NULL`

Referenced by `virus_engine_create()`.

5.412.2.31 `int(* cl_init_d)(unsigned int initoptions) = NULL`

Referenced by `virus_start()`.

5.412.2.32 `int(* cl_load_d)(const char *path, struct cl_engine *engine, unsigned int *signo, unsigned int dboptions) = NULL`

Referenced by `virus_engine_create()`.

5.412.2.33 `const char*(* cl_retver_d)(void) = NULL`

Referenced by `lib_version_clamav()`.

5.412.2.34 `int(* cl_scandesc_d)(int desc, const char **virname, unsigned long int *scanned, const struct cl_engine *engine, unsigned int scanoptions) = NULL`

Referenced by `virus_check()`.

5.412.2.35 `void(* cl_shutdown_d)(void) = NULL`

CLAMAV.

Referenced by `virus_stop()`.

5.412.2.36 `int(* cl_statchkdir_d)(const struct cl_stat *dbstat) = NULL`

Referenced by `virus_engine_refresh()`.

5.412.2.37 `int(* cl_statfree_d)(struct cl_stat *dbstat) = NULL`

Referenced by `virus_engine_refresh()`, `virus_start()`, and `virus_stop()`.

5.412.2.38 `int(* cl_statinidir_d)(const char *dirname, struct cl_stat *dbstat) = NULL`

Referenced by `virus_engine_refresh()`, and `virus_start()`.

5.412.2.39 `const char*(* cl_strerror_d)(int clerror) = NULL`

Referenced by `virus_check()`, `virus_engine_create()`, `virus_sigs_total()`, and `virus_start()`.

5.412.2.40 `void(* COMP_zlib_cleanup_d)(void) = NULL`

Referenced by `ssl_stop()`.

5.412.2.41 void(* CONF_modules_unload_d)(int all) = NULL

Referenced by ssl_stop().

5.412.2.42 void(* CRYPTO_cleanup_all_ex_data_d)(void) = NULL

Referenced by ssl_stop().

5.412.2.43 void(* CRYPTO_free_d)(void *) = NULL

Referenced by _get_cache_ocsp_id(), and _get_x509_cert_sha_hash().

5.412.2.44 int(* CRYPTO_num_locks_d)(void) = NULL

Referenced by ssl_start(), and ssl_stop().

5.412.2.45 int(* DH_check_d)(const DH *dh, int *ret) = NULL

Referenced by dh_exchange_2048(), dh_exchange_4096(), and dh_params_generate().

5.412.2.46 void(* DH_free_d)(DH *dh) = NULL

Referenced by dh_exchange_2048(), dh_exchange_4096(), dh_params_2048(), dh_params_4096(), and dh_params_generate().

5.412.2.47 DH*(* DH_new_d)(void) = NULL

OPENSSL.

Referenced by dh_exchange_2048(), dh_exchange_4096(), dh_params_2048(), dh_params_4096(), and dh_params_generate().

5.412.2.48 DKIM_STAT(* dkim_body_d)(DKIM *dkim, u_char *buf, size_t len) = NULL

5.412.2.49 DKIM_STAT(* dkim_chunk_d)(DKIM *dkim, unsigned char *chunkp, size_t len) = NULL

Referenced by dkim_signature_create(), and dkim_signature_verify().

5.412.2.50 void(* dkim_close_d)(DKIM_LIB *lib) = NULL

Referenced by dkim_stop().

5.412.2.51 DKIM_STAT(* dkim_eoh_d)(DKIM *dkim) = NULL

DKIM

Note:

that dkim_getsighdr_d is used by the library, so were using dkim_getsighdrx_d.

5.412.2.52 DKIM_STAT(* dkim_eom_d)(DKIM *dkim, _Bool *testkey) = NULL

Referenced by dkim_signature_create(), and dkim_signature_verify().

5.412.2.53 `DKIM_STAT(* dkim_free_d)(DKIM *dkim) = NULL`

Referenced by `dkim_signature_create()`, and `dkim_signature_verify()`.

5.412.2.54 `char>(* dkim_geterror_d)(DKIM *dkim) = NULL`

Referenced by `dkim_signature_create()`.

5.412.2.55 `const char>(* dkim_getresultstr_d)(DKIM_STAT result) = NULL`

Referenced by `dkim_signature_create()`, and `dkim_signature_verify()`.

5.412.2.56 `DKIM_STAT(* dkim_getsighdrx_d)(DKIM *dkim, u_char *buf, size_t len, size_t initial) = NULL`

Referenced by `dkim_signature_create()`, and `lib_load_dkim()`.

5.412.2.57 `DKIM_STAT(* dkim_header_d)(DKIM *dkim, u_char *hdr, size_t len) = NULL`

5.412.2.58 `DKIM_LIB(* dkim_init_d)(void *(*mallocf)(void *closure, size_t nbytes), void(*freef)(void *closure, void *p)) = NULL`

Referenced by `dkim_start()`.

5.412.2.59 `uint32_t(* dkim_libversion_d)(void) = NULL`

Referenced by `lib_load_dkim()`.

5.412.2.60 `void(* dkim_mfree_d)(DKIM_LIB *libhandle, void *closure, void *ptr) = NULL`

5.412.2.61 `DKIM(* dkim_sign_d)(DKIM_LIB *libhandle, const unsigned char *id, void *memclosure, const dkim_sigkey_t secretkey, const unsigned char *selector, const unsigned char *domain, dkim_canon_t hdr_canon_alg, dkim_canon_t body_canon_alg, dkim_alg_t sign_alg, off_t length, DKIM_STAT *statp) = NULL`

Referenced by `dkim_signature_create()`.

5.412.2.62 `int(* dkim_test_dns_put_d)(DKIM *dkim, int class, int type, int prec, u_char *name, u_char *data) = NULL`

5.412.2.63 `DKIM(* dkim_verify_d)(DKIM_LIB *libhandle, const unsigned char *id, void *memclosure, DKIM_STAT *statp) = NULL`

Referenced by `dkim_signature_verify()`.

5.412.2.64 `int(* dspam_attach_d)(DSPAM_CTX *CTX, void *dbh) = NULL`

Referenced by `dspam_check()`, and `dspam_train()`.

5.412.2.65 `DSPAM_CTX(* dspam_create_d)(const char *username, const char *group, const char *home, int operating_mode, u_int32_t flags) = NULL`

Referenced by `dspam_check()`, and `dspam_train()`.

5.412.2.66 `void(* dspam_destroy_d)(DSPAM_CTX *CTX) = NULL`

Referenced by `dspam_check()`, and `dspam_train()`.

5.412.2.67 `int(* dspam_detach_d)(DSPAM_CTX *CTX) = NULL`

Referenced by `dspam_check()`, and `dspam_train()`.

5.412.2.68 `int(* dspam_init_driver_d)(DRIVER_CTX *DTX) = NULL`

Referenced by `dspam_start()`.

5.412.2.69 `int(* dspam_process_d)(DSPAM_CTX *CTX, const char *message) = NULL`

Referenced by `dspam_check()`, and `dspam_train()`.

5.412.2.70 `int(* dspam_shutdown_driver_d)(DRIVER_CTX *DTX) = NULL`

Referenced by `dspam_stop()`.

5.412.2.71 `const char*(* dspam_version_d)(void) = NULL`

DSPAM.

Referenced by `lib_version_dspam()`.

5.412.2.72 `void(* EC_GROUP_clear_free_d)(EC_GROUP *group) = NULL`

Referenced by `_crypto_shutdown()`, `encrypt_ctx_free()`, and `encrypt_ctx_new()`.

5.412.2.73 `void(* EC_GROUP_free_d)(EC_GROUP *group) = NULL`

Referenced by `deprecated_ecies_group()`, `deprecated_ecies_stop()`, `ecies_group()`, `ecies_stop()`, `prime_start()`, and `prime_stop()`.

5.412.2.74 `EC_GROUP*(* EC_GROUP_new_by_curve_name_d)(int nid) = NULL`

Referenced by `_crypto_init()`, `_deserialize_ec_pubkey()`, `deprecated_ecies_group()`, `ecies_group()`, `encrypt_ctx_new()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, and `prime_start()`.

5.412.2.75 `int(* EC_GROUP_precompute_mult_d)(EC_GROUP *group, BN_CTX *ctx) = NULL`

Referenced by `deprecated_ecies_group()`, `ecies_group()`, and `prime_start()`.

5.412.2.76 `int(* EC_KEY_check_key_d)(const EC_KEY *key) = NULL`

Referenced by `deprecated_ecies_key_public()`, `ecies_key_public()`, and `secp256k1_private_set()`.

5.412.2.77 void(* EC_KEY_free_d)(EC_KEY *key) = NULL

Referenced by `_deserialize_ec_pubkey()`, `_free_ec_key()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_key_alloc()`, `deprecated_ecies_key_create()`, `deprecated_ecies_key_free()`, `deprecated_ecies_key_private()`, `deprecated_ecies_key_public()`, `ecies_decrypt()`, `ecies_encrypt()`, `ecies_key_alloc()`, `ecies_key_create()`, `ecies_key_free()`, `ecies_key_private()`, `ecies_key_public()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, `encrypt_keypair_generate()`, `secp256k1_alloc()`, `secp256k1_free()`, `secp256k1_generate()`, `secp256k1_private_set()`, `secp256k1_public_set()`, and `ssl_stop()`.

5.412.2.78 int(* EC_KEY_generate_key_d)(EC_KEY *key) = NULL

Referenced by `deprecated_ecies_key_create()`, `ecies_key_create()`, `encrypt_keypair_generate()`, and `secp256k1_generate()`.

5.412.2.79 const EC_GROUP*(* EC_KEY_get0_group_d)(const EC_KEY *key) = NULL

Referenced by `deprecated_ecies_encrypt()`, `deprecated_ecies_key_public()`, `deprecated_ecies_key_public_bin()`, `deprecated_ecies_key_public_hex()`, `ecies_encrypt()`, `ecies_key_public()`, `ecies_key_public_bin()`, `ecies_key_public_hex()`, `secp256k1_private_set()`, `secp256k1_public_get()`, `secp256k1_public_set()`, and `ssl_ecdh_exchange_callback()`.

5.412.2.80 const BIGNUM*(* EC_KEY_get0_private_key_d)(const EC_KEY *key) = NULL

Referenced by `deprecated_ecies_key_private_bin()`, `deprecated_ecies_key_private_hex()`, `ecies_key_private_bin()`, `ecies_key_private_hex()`, `secp256k1_private_get()`, and `secp256k1_type()`.

5.412.2.81 const EC_POINT*(* EC_KEY_get0_public_key_d)(const EC_KEY *key) = NULL

Referenced by `_compute_aes256_kek()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_key_public_bin()`, `deprecated_ecies_key_public_hex()`, `ecies_decrypt()`, `ecies_encrypt()`, `ecies_key_public_bin()`, `ecies_key_public_hex()`, `secp256k1_compute_kek()`, `secp256k1_public_get()`, and `secp256k1_type()`.

5.412.2.82 EC_KEY*(* EC_KEY_new_by_curve_name_d)(int nid) = NULL

Referenced by `deprecated_ecies_key_alloc()`, `ecies_key_alloc()`, `secp256k1_alloc()`, and `ssl_ecdh_exchange_callback()`.

5.412.2.83 EC_KEY*(* EC_KEY_new_d)(void) = NULL

Referenced by `_deserialize_ec_pubkey()`, `deprecated_ecies_key_alloc()`, `ecies_key_alloc()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, `encrypt_keypair_generate()`, and `secp256k1_alloc()`.

5.412.2.84 int(* EC_KEY_set_group_d)(EC_KEY *key, const EC_GROUP *group) = NULL

Referenced by `_deserialize_ec_pubkey()`, `deprecated_ecies_key_alloc()`, `ecies_key_alloc()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, `encrypt_keypair_generate()`, and `secp256k1_alloc()`.

5.412.2.85 int(* EC_KEY_set_private_key_d)(EC_KEY *key, const BIGNUM *prv) = NULL

Referenced by `deprecated_ecies_key_private()`, `ecies_key_private()`, and `secp256k1_private_set()`.

5.412.2.86 void(* EC_POINT_free_d)(EC_POINT *point) = NULL

Referenced by `deprecated_ecies_key_public()`, `ecies_key_public()`, `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.412.2.87 `EC_POINT*(* EC_POINT_new_d)(const EC_GROUP *group) = NULL`

Referenced by `deprecated_ecies_key_public()`, `ecies_key_public()`, `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.412.2.88 `void(* ECDSA_SIG_free_d)(ECDSA_SIG *a) = NULL`

Referenced by `_ec_sign_data()`, and `_verify_ec_signature()`.

5.412.2.89 `void(* ENGINE_cleanup_d)(void) = NULL`

Referenced by `ssl_stop()`.

5.412.2.90 `void(* ERR_clear_error_d)(void) = NULL`

Referenced by `tls_client_alloc()`, `tls_read()`, `tls_server_alloc()`, and `tls_write()`.

5.412.2.91 `void(* ERR_free_strings_d)(void) = NULL`

Referenced by `_crypto_shutdown()`, `_ssl_shutdown()`, `encrypt_ctx_free()`, `encrypt_ctx_new()`, and `ssl_stop()`.

5.412.2.92 `unsigned long(* ERR_get_error_d)(void) = NULL`

Referenced by `_push_error_stack_openssl()`, `ssl_error_string()`, `tls_continue()`, and `tls_error()`.

5.412.2.93 `void(* ERR_load_crypto_strings_d)(void) = NULL`

Referenced by `_push_error_stack_openssl()`.

5.412.2.94 `unsigned long(* ERR_peek_error_d)(void) = NULL`

5.412.2.95 `void(* ERR_print_errors_fp_d)(FILE *fp) = NULL`

Referenced by `_do_ocsp_validation()`, `_dump_ocsp_response_cb()`, `_ocsp_response_callback()`, `_ssl_disconnect()`, `_ssl_fd_loop()`, and `_verify_certificate_callback()`.

5.412.2.96 `void(* ERR_remove_thread_state_d)(const CRYPTO_THREADID *tid) = NULL`

Referenced by `ssl_stop()`, `ssl_thread_stop()`, and `tls_free()`.

5.412.2.97 `const EVP_CIPHER*(* EVP_aes_256_cbc_d)(void) = NULL`

Referenced by `_decrypt_aes_256()`, and `_encrypt_aes_256()`.

5.412.2.98 `const EVP_CIPHER*(* EVP_aes_256_gcm_d)(void) = NULL`

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `stacie_decrypt()`, and `stacie_encrypt()`.

5.412.2.99 `int(* EVP_CIPHER_block_size_d)(const EVP_CIPHER *cipher) = NULL`

Referenced by `cipher_block_length()`, `deprecated_ecies_encrypt()`, and `ecies_encrypt()`.

5.412.2.100 `int(* EVP_CIPHER_CTX_block_size_d)(const EVP_CIPHER_CTX *ctx) = NULL`

Referenced by `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `symmetric_decrypt()`, and `symmetric_encrypt()`.

5.412.2.101 `int(* EVP_CIPHER_CTX_cleanup_d)(EVP_CIPHER_CTX *a) = NULL`

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `ecies_decrypt()`, `ecies_encrypt()`, `stacie_decrypt()`, `stacie_encrypt()`, `symmetric_decrypt()`, and `symmetric_encrypt()`.

5.412.2.102 `void(* EVP_CIPHER_CTX_free_d)(EVP_CIPHER_CTX *a) = NULL`

Referenced by `_decrypt_aes_256()`, and `_encrypt_aes_256()`.

5.412.2.103 `void(* EVP_CIPHER_CTX_init_d)(EVP_CIPHER_CTX *a) = NULL`

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `ecies_decrypt()`, `ecies_encrypt()`, `stacie_decrypt()`, `stacie_encrypt()`, `symmetric_decrypt()`, and `symmetric_encrypt()`.

5.412.2.104 `int(* EVP_CIPHER_CTX_iv_length_d)(const EVP_CIPHER_CTX *ctx) = NULL`

Referenced by `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `symmetric_decrypt()`, and `symmetric_encrypt()`.

5.412.2.105 `int(* EVP_CIPHER_CTX_key_length_d)(const EVP_CIPHER_CTX *ctx) = NULL`

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `stacie_decrypt()`, `stacie_encrypt()`, `symmetric_decrypt()`, and `symmetric_encrypt()`.

5.412.2.106 `EVP_CIPHER_CTX*(* EVP_CIPHER_CTX_new_d)(void) = NULL`

Referenced by `_decrypt_aes_256()`, and `_encrypt_aes_256()`.

5.412.2.107 `int(* EVP_CIPHER_CTX_set_padding_d)(EVP_CIPHER_CTX *c, int pad) = NULL`

Referenced by `_decrypt_aes_256()`, `_encrypt_aes_256()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `ecies_decrypt()`, and `ecies_encrypt()`.

5.412.2.108 `unsigned long(* EVP_CIPHER_flags_d)(const EVP_CIPHER *cipher) = NULL`

Referenced by `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `symmetric_decrypt()`, and `symmetric_encrypt()`.

5.412.2.109 `int(* EVP_CIPHER_iv_length_d)(const EVP_CIPHER *cipher) = NULL`

Referenced by `cipher_vector_length()`.

5.412.2.110 `int(* EVP_CIPHER_key_length_d)(const EVP_CIPHER *cipher) = NULL`

Referenced by `cipher_key_length()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `ecies_decrypt()`, and `ecies_encrypt()`.

5.412.2.111 `int(* EVP_CIPHER_nid_d)(const EVP_CIPHER *cipher) = NULL`

Referenced by `cipher_block_length()`, `cipher_key_length()`, `cipher_numeric_id()`, and `cipher_vector_length()`.

5.412.2.112 `void(* EVP_cleanup_d)(void) = NULL`

Referenced by `_crypto_shutdown()`, `encrypt_ctx_free()`, `encrypt_ctx_new()`, and `ssl_stop()`.

5.412.2.113 `int(* EVP_DigestInit_d)(EVP_MD_CTX *ctx, const EVP_MD *type) = NULL`

Referenced by `_rsa_verify_record()`.

5.412.2.114 `const EVP_CIPHER*(* EVP_get_cipherbyname_d)(const char *name) = NULL`

Referenced by `cipher_id()`, `cipher_name()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_start()`, `ecies_decrypt()`, `ecies_encrypt()`, and `ecies_start()`.

5.412.2.115 `const EVP_MD*(* EVP_get_digestbyname_d)(const char *name) = NULL`

Referenced by `_crypto_init()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_start()`, `digest_id()`, `digest_name()`, `ecies_decrypt()`, `ecies_encrypt()`, `ecies_start()`, and `encrypt_ctx_new()`.

5.412.2.116 `const EVP_MD*(* EVP_md4_d)(void) = NULL`

Referenced by `deprecated_hmac_md4()`, `hash_md4()`, and `hmac_md4()`.

5.412.2.117 `const EVP_MD*(* EVP_md5_d)(void) = NULL`

Referenced by `deprecated_hmac_md5()`, `hash_md5()`, and `hmac_md5()`.

5.412.2.118 `int(* EVP_MD_CTX_cleanup_d)(EVP_MD_CTX *ctx) = NULL`

Referenced by `hash_digest()`, `stacie_derive_key()`, `stacie_derive_token()`, and `stacie_realm_key()`.

5.412.2.119 `void(* EVP_MD_CTX_init_d)(EVP_MD_CTX *ctx) = NULL`

Referenced by `_rsa_verify_record()`, `hash_digest()`, `stacie_derive_key()`, `stacie_derive_token()`, and `stacie_realm_key()`.

5.412.2.120 `int(* EVP_MD_size_d)(const EVP_MD *md) = NULL`

Referenced by `deprecated_ecies_encrypt()`, `deprecated_hmac_digest()`, `digest_length_output()`, `ecies_encrypt()`, `hash_digest()`, and `hmac_digest()`.

5.412.2.121 `int(* EVP_MD_type_d)(const EVP_MD *md) = NULL`

Referenced by `digest_length_output()`.

5.412.2.122 `EVP_PKEY*(* EVP_PKEY_new_d)(void) = NULL`

Referenced by `_rsa_verify_record()`.

5.412.2.123 `int(* EVP_PKEY_set1_RSA_d)(EVP_PKEY *pkey, struct rsa_st *key) = NULL`

Referenced by `_rsa_verify_record()`.

5.412.2.124 `const EVP_MD*(* EVP_ripemd160_d)(void) = NULL`

Referenced by `deprecated_hmac_ripemd160()`, `hash_ripemd160()`, and `hmac_ripemd160()`.

5.412.2.125 `const EVP_MD*(* EVP_sha1_d)(void) = NULL`

Referenced by `_rsa_verify_record()`, `deprecated_hmac_sha1()`, `hash_sha1()`, and `hmac_sha1()`.

5.412.2.126 `const EVP_MD*(* EVP_sha224_d)(void) = NULL`

Referenced by `deprecated_hmac_sha224()`, `hash_sha224()`, and `hmac_sha224()`.

5.412.2.127 `const EVP_MD*(* EVP_sha256_d)(void) = NULL`

Referenced by `_rsa_verify_record()`, `deprecated_hmac_sha256()`, `hash_sha256()`, and `hmac_sha256()`.

5.412.2.128 `const EVP_MD*(* EVP_sha384_d)(void) = NULL`

Referenced by `deprecated_hmac_sha384()`, `hash_sha384()`, and `hmac_sha384()`.

5.412.2.129 `const EVP_MD*(* EVP_sha512_d)(void) = NULL`

Referenced by `_rsa_verify_record()`, `deprecated_hmac_sha512()`, `hash_sha512()`, `hmac_sha512()`, `stacie_derive_key()`, `stacie_derive_seed()`, `stacie_derive_token()`, and `stacie_realms_key()`.

5.412.2.130 `const EVP_MD*(* EVP_sha_d)(void) = NULL`

Referenced by `deprecated_hmac_sha()`, `hash_sha()`, and `hmac_sha()`.

5.412.2.131 `FT_Error(* FT_Done_FreeType_d)(FT_Library library) = NULL`

FreeType.

Referenced by `lib_version_freetype()`.

5.412.2.132 `FT_Error(* FT_Init_FreeType_d)(FT_Library *alibrary) = NULL`

Referenced by `lib_version_freetype()`.

5.412.2.133 `void(* FT_Library_Version_d)(FT_Library library, FT_Int *amajor, FT_Int *aminor, FT_Int *apatch) = NULL`

Referenced by `lib_version_freetype()`.

5.412.2.134 `void(* gdFree_d)(void *m) = NULL`

Referenced by `register_captcha_generate()`.

5.412.2.135 `int(* gdImageColorResolve_d)(gdImagePtr im, int r, int g, int b) = NULL`

Referenced by `register_captcha_generate()`, and `register_captcha_write_noise()`.

5.412.2.136 `gdImagePtr(* gdImageCreate_d)(int sx, int sy) = NULL`

Referenced by `register_captcha_generate()`.

5.412.2.137 `void(* gdImageDestroy_d)(gdImagePtr im) = NULL`

Referenced by `register_captcha_generate()`.

5.412.2.138 `void(* gdImageGifPtr_d)(gdImagePtr im, int *size) = NULL`

Referenced by `register_captcha_generate()`.

5.412.2.139 `void(* gdImageJpegPtr_d)(gdImagePtr im, int *size, int quality) = NULL`

5.412.2.140 `void(* gdImageSetPixel_d)(gdImagePtr im, int x, int y, int color) = NULL`

Referenced by `register_captcha_write_noise()`.

5.412.2.141 `char(* gdImageStringFT_d)(gdImage *im, int *breect, int fg, char *fontlist, double psize, double angle, int x, int y, char *string) = NULL`

Referenced by `register_captcha_generate()`.

5.412.2.142 `const char(* gdVersionString_d)(void) = NULL`

GD.

Referenced by `lib_version_gd()`.

5.412.2.143 `void(* HMAC_CTX_cleanup_d)(HMAC_CTX *ctx) = NULL`

Referenced by `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_hmac_digest()`, `ecies_decrypt()`, `ecies_encrypt()`, `hmac_digest()`, and `stacie_derive_seed()`.

5.412.2.144 `void(* HMAC_CTX_init_d)(HMAC_CTX *ctx) = NULL`

Referenced by `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_hmac_digest()`, `ecies_decrypt()`, `ecies_encrypt()`, `hmac_digest()`, and `stacie_derive_seed()`.

5.412.2.145 `int(* i2d_ECPrivateKey_d)(EC_KEY *key, unsigned char **out) = NULL`

5.412.2.146 `int(* i2d_OCSP_CERTID_d)(OCSP_CERTID *a, unsigned char **out) = NULL`

Referenced by `_get_cache_ocsp_id()`.

5.412.2.147 `int(* i2d_X509_d)(X509 *a, unsigned char **out) = NULL`

Referenced by `_get_x509_cert_sha_hash()`.

5.412.2.148 `int(* i2o_ECPublicKey_d)(EC_KEY *key, unsigned char **out) = NULL`

5.412.2.149 `const char*(* jpeg_version_d)(void) = NULL`

JPEG.

Referenced by `lib_version_jpeg()`.

5.412.2.150 `int(* lt_dlexit_d)(void) = NULL`

5.412.2.151 `int(* lzo1x_1_compress_d)(const lzo_byte *src, lzo_uint src_len, lzo_byte *dst, lzo_uintp dst_len, lzo_voidp wrkmem) = NULL`

Referenced by `compress_lzo()`.

5.412.2.152 `int(* lzo1x_decompress_safe_d)(const lzo_byte *src, lzo_uint src_len, lzo_byte *dst, lzo_uintp dst_len, lzo_voidp wrkmem) = NULL`

Referenced by `decompress_block_lzo()`, and `decompress_lzo()`.

5.412.2.153 `lzo_uint32(* lzo_adler32_d)(lzo_uint32 _adler, const lzo_bytew _buf, lzo_uint _len) = NULL`

5.412.2.154 `const char*(* lzo_version_string_d)(void) = NULL`

LZO.

Referenced by `lib_version_lzo()`.

5.412.2.155 `memcached_return_t(* memcached_add_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL`

Referenced by `cache_add()`, `cache_append()`, and `cache_silent_add()`.

5.412.2.156 `memcached_return_t(* memcached_append_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL`

Referenced by `cache_append()`.

5.412.2.157 `memcached_return_t(* memcached_behavior_set_d)(memcached_st *ptr, const memcached_behavior_t flag, uint64_t data) = NULL`

Referenced by `cache_start()`.

- 5.412.2.158** `memcached_return_t(* memcached_cas_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags, uint64_t cas) = NULL`
- 5.412.2.159** `memcached_st(* memcached_create_d)(memcached_st *ptr) = NULL`

Referenced by `cache_start()`.

- 5.412.2.160** `memcached_return_t(* memcached_decrement_d)(memcached_st *ptr, const char *key, size_t key_length, uint32_t offset, uint64_t *value) = NULL`
- 5.412.2.161** `memcached_return_t(* memcached_decrement_with_initial_d)(memcached_st *ptr, const char *key, size_t key_length, uint64_t offset, uint64_t initial, time_t expiration, uint64_t *value) = NULL`

Referenced by `cache_decrement()`.

- 5.412.2.162** `memcached_return_t(* memcached_delete_d)(memcached_st *ptr, const char *key, size_t key_length, time_t expiration) = NULL`

Referenced by `cache_delete()`.

- 5.412.2.163** `memcached_return_t(* memcached_flush_d)(memcached_st *ptr, time_t expiration) = NULL`

MEMCACHED.

Referenced by `cache_flush()`.

- 5.412.2.164** `void(* memcached_free_d)(memcached_st *ptr) = NULL`

Referenced by `cache_start()`, and `cache_stop()`.

- 5.412.2.165** `char(* memcached_get_d)(memcached_st *ptr, const char *key, size_t key_length, size_t *value_length, uint32_t *flags, memcached_return_t *error) = NULL`

Referenced by `cache_get()`, and `cache_get_u64()`.

- 5.412.2.166** `memcached_return_t(* memcached_increment_d)(memcached_st *ptr, const char *key, size_t key_length, uint32_t offset, uint64_t *value) = NULL`

- 5.412.2.167** `memcached_return_t(* memcached_increment_with_initial_d)(memcached_st *ptr, const char *key, size_t key_length, uint64_t offset, uint64_t initial, time_t expiration, uint64_t *value) = NULL`

Referenced by `cache_increment()`.

- 5.412.2.168** `const char(* memcached_lib_version_d)(void) = NULL`

Referenced by `lib_version_cache()`.

5.412.2.169 `memcached_return_t(* memcached_prepend_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL`

5.412.2.170 `memcached_return_t(* memcached_replace_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL`

5.412.2.171 `memcached_return_t(* memcached_server_add_with_weight_d)(memcached_st *ptr, const char *hostname, in_port_t port, uint32_t weight) = NULL`

Referenced by `cache_start()`.

5.412.2.172 `memcached_return_t(* memcached_set_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags) = NULL`

Referenced by `cache_set()`, and `cache_set_u64()`.

5.412.2.173 `const char*(* memcached_strerror_d)(const memcached_st *ptr, memcached_return_t rc) = NULL`

Referenced by `cache_add()`, `cache_append()`, `cache_decrement()`, `cache_delete()`, `cache_flush()`, `cache_get()`, `cache_get_u64()`, `cache_increment()`, `cache_set()`, `cache_set_u64()`, and `cache_start()`.

5.412.2.174 `void(* my_once_free_d)(void) = NULL`

MYSQL.

Referenced by `sql_stop()`.

5.412.2.175 `my_ulonglong(* mysql_affected_rows_d)(MYSQL *mysql) = NULL`

Referenced by `sql_write_conn()`.

5.412.2.176 `const char*(* mysql_character_set_name_d)(MYSQL *mysql) = NULL`

Referenced by `serv_charset_mysql()`.

5.412.2.177 `void(* mysql_close_d)(MYSQL *mysql) = NULL`

Referenced by `serv_charset_mysql()`, `serv_schema_mysql()`, `serv_version_mysql()`, `sql_open()`, and `sql_stop()`.

5.412.2.178 `my_bool(* mysql_embedded_d)(void) = NULL`

Referenced by `serv_type_mysql()`.

5.412.2.179 `unsigned int(* mysql_errno_d)(MYSQL *mysql) = NULL`

Referenced by `sql_errno()`, `sql_error()`, `stmt_errno()`, and `stmt_error()`.

5.412.2.180 `const char*(* mysql_error_d)(MYSQL *mysql) = NULL`

Referenced by `sql_error()`, `sql_insert_conn()`, `sql_num_rows_conn()`, `sql_query_res_conn()`, `sql_write_conn()`, and `stmt_error()`.

5.412.2.181 `unsigned long(* mysql_escape_string_d)(char *to, const char *from, unsigned long length) = NULL`

5.412.2.182 `MYSQL_FIELD*(* mysql_fetch_field_d)(MYSQL_RES *result) = NULL`

Referenced by `res_bind_create()`.

5.412.2.183 `MYSQL_ROW(* mysql_fetch_row_d)(MYSQL_RES *result) = NULL`

5.412.2.184 `void(* mysql_free_result_d)(MYSQL_RES *result) = NULL`

Referenced by `res_bind_create()`, and `sql_num_rows_conn()`.

5.412.2.185 `unsigned long(* mysql_get_client_version_d)(void) = NULL`

Referenced by `lib_version_mysql()`.

5.412.2.186 `const char*(* mysql_get_server_info_d)(MYSQL *mysql) = NULL`

Referenced by `serv_version_mysql()`.

5.412.2.187 `MYSQL*(* mysql_init_d)(MYSQL *mysql) = NULL`

Referenced by `sql_open()`.

5.412.2.188 `my_ulonglong(* mysql_insert_id_d)(MYSQL *mysql) = NULL`

Referenced by `sql_insert_conn()`.

5.412.2.189 `unsigned int(* mysql_num_fields_d)(MYSQL_RES *result) = NULL`

Referenced by `res_bind_create()`.

5.412.2.190 `my_ulonglong(* mysql_num_rows_d)(MYSQL_RES *result) = NULL`

Referenced by `sql_num_rows_conn()`.

5.412.2.191 `int(* mysql_options_d)(MYSQL *mysql, enum mysql_option option, const void *arg) = NULL`

Referenced by `sql_open()`.

5.412.2.192 `int(* mysql_ping_d)(MYSQL *mysql) = NULL`

Referenced by `sql_ping()`.

5.412.2.193 `MYSQL*(* mysql_real_connect_d)(MYSQL *mysql, const char *name, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag) = NULL`

Referenced by `sql_open()`.

5.412.2.194 `int(* mysql_real_query_d)(MYSQL *mysql, const char *query, unsigned long length) = NULL`

Referenced by `sql_insert_conn()`, `sql_num_rows_conn()`, `sql_query_conn()`, `sql_query_res_conn()`, and `sql_write_conn()`.

5.412.2.195 `void(* mysql_server_end_d)(void) = NULL`

Referenced by `sql_stop()`.

5.412.2.196 `int(* mysql_server_init_d)(int argc, char **argv, char **groups) = NULL`

Referenced by `sql_start()`.

5.412.2.197 `int(* mysql_set_character_set_d)(MYSQL *mysql, const char *csname) = NULL`

5.412.2.198 `my_ulonglong(* mysql_stmt_affected_rows_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `stmt_exec_affected_conn()`.

5.412.2.199 `my_bool(* mysql_stmt_attr_set_d)(MYSQL_STMT *stmt, enum enum_stmt_attr_type attr_type, const void *attr) = NULL`

Referenced by `stmt_prepare()`.

5.412.2.200 `my_bool(* mysql_stmt_bind_param_d)(MYSQL_STMT *stmt, MYSQL_BIND *bind) = NULL`

Referenced by `stmt_bind_param()`.

5.412.2.201 `my_bool(* mysql_stmt_bind_result_d)(MYSQL_STMT *stmt, MYSQL_BIND *bind) = NULL`

Referenced by `res_bind_free()`, and `res_stmt_store()`.

5.412.2.202 `my_bool(* mysql_stmt_close_d)(MYSQL_STMT *) = NULL`

Referenced by `stmt_close()`.

5.412.2.203 `unsigned int(* mysql_stmt_errno_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `mail_db_update_message_folder()`, `stmt_errno()`, and `stmt_error()`.

5.412.2.204 `const char*(* mysql_stmt_error_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `mail_db_update_message_folder()`, `res_bind_create()`, `res_stmt_store()`, and `stmt_error()`.

5.412.2.205 `int(* mysql_stmt_execute_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `stmt_exec_affected_conn()`, `stmt_exec_conn()`, `stmt_get_result_conn()`, and `stmt_insert_conn()`.

5.412.2.206 `int(* mysql_stmt_fetch_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `res_stmt_store()`.

5.412.2.207 `my_bool(* mysql_stmt_free_result_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `res_stmt_store()`.

5.412.2.208 `MYSQL_STMT*(* mysql_stmt_init_d)(MYSQL *mysql) = NULL`

Referenced by `stmt_open()`.

5.412.2.209 `my_ulonglong(* mysql_stmt_insert_id_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `stmt_insert_conn()`.

5.412.2.210 `my_ulonglong(* mysql_stmt_num_rows_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `res_stmt_store()`.

5.412.2.211 `int(* mysql_stmt_prepare_d)(MYSQL_STMT *stmt, const char *query, unsigned long length) = NULL`

Referenced by `stmt_prepare()`.

5.412.2.212 `my_bool(* mysql_stmt_reset_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `stmt_reset()`.

5.412.2.213 `MYSQL_RES*(* mysql_stmt_result_metadata_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `res_bind_create()`.

5.412.2.214 `int(* mysql_stmt_store_result_d)(MYSQL_STMT *stmt) = NULL`

Referenced by `res_stmt_store()`.

5.412.2.215 `MYSQL_RES*(* mysql_store_result_d)(MYSQL *mysql) = NULL`

Referenced by `sql_num_rows_conn()`, and `sql_query_res_conn()`.

5.412.2.216 `void(* mysql_thread_end_d)(void) = NULL`

Referenced by `sql_thread_stop()`.

5.412.2.217 `unsigned long(* mysql_thread_id_d)(MYSQL *mysql) = NULL`

Referenced by `sql_ping()`.

5.412.2.218 `my_bool(* mysql_thread_init_d)(void) = NULL`

Referenced by `sql_thread_start()`.

5.412.2.219 `unsigned int(* mysql_thread_safe_d)(void) = NULL`

Referenced by `sql_start()`.

5.412.2.220 `void(* OBJ_cleanup_d)(void) = NULL`

Referenced by `ssl_stop()`.

5.412.2.221 `void(* OBJ_NAME_cleanup_d)(int type) = NULL`

Referenced by `ssl_stop()`.

5.412.2.222 `const char*(* OBJ_nid2sn_d)(int n) = NULL`

Referenced by `_crypto_init()`, `cipher_block_length()`, `cipher_id()`, `cipher_key_length()`, `cipher_vector_length()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_start()`, `digest_id()`, `digest_length_output()`, `ecies_decrypt()`, `ecies_encrypt()`, `ecies_start()`, and `encrypt_ctx_new()`.

5.412.2.223 `void(* OCSP_BASICRESP_free_d)(OCSP_BASICRESP *a) = NULL`

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.412.2.224 `int(* OCSP_check_nonce_d)(void *req, void *bs) = NULL`

Referenced by `_do_ocsp_validation()`.

5.412.2.225 `int(* OCSP_REQ_CTX_set1_req_d)(OCSP_REQ_CTX *rctx, void *req) = NULL`

Referenced by `_do_ocsp_validation()`.

5.412.2.226 `void(* OCSP_request_add0_id_d)(void *req, void *cid) = NULL`

Referenced by `_do_ocsp_validation()`.

5.412.2.227 `void(* OCSP_REQUEST_free_d)(OCSP_REQUEST *a) = NULL`

Referenced by `_do_ocsp_validation()`.

5.412.2.228 `OCSP_REQUEST*(* OCSP_REQUEST_new_d)(void) = NULL`

Referenced by `_do_ocsp_validation()`.

5.412.2.229 `void(* OCSP_RESPONSE_free_d)(OCSP_RESPONSE *a) = NULL`

Referenced by `_destroy_ocsp_response_cb()`, `_do_ocsp_validation()`, and `_ocsp_response_callback()`.

5.412.2.230 `void>(* OCSP_response_get1_basic_d)(OCSP_RESPONSE *resp) = NULL`

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.412.2.231 `int(* OCSP_response_status_d)(OCSP_RESPONSE *resp) = NULL`

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.412.2.232 `const char>(* OCSP_response_status_str_d)(long s) = NULL`

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.412.2.233 `void(* OPENSSL_add_all_algorithms_noconf_d)(void) = NULL`

Referenced by `_crypto_init()`, `encrypt_ctx_new()`, and `ssl_start()`.

5.412.2.234 `int(* RAND_bytes_d)(unsigned char *buf, int num) = NULL`

Referenced by `_generate_ed25519_keypair()`, `_get_random_bytes()`, `ed25519_randombytes_unsafe()`, `rand_choices()`, `rand_get_int16()`, `rand_get_int32()`, `rand_get_int64()`, `rand_get_int8()`, `rand_get_uint16()`, `rand_get_uint32()`, `rand_get_uint64()`, `rand_get_uint8()`, and `rand_write()`.

5.412.2.235 `void(* RAND_cleanup_d)(void) = NULL`

Referenced by `rand_stop()`.

5.412.2.236 `int(* RAND_load_file_d)(const char *filename, long max_bytes) = NULL`

Referenced by `rand_start()`.

5.412.2.237 `int(* RAND_status_d)(void) = NULL`

Referenced by `rand_start()`.

5.412.2.238 `void(* RSA_free_d)(RSA *r) = NULL`

Referenced by `_destroy_dnskey()`.

5.412.2.239 `RSA>(* RSA_new_d)(void) = NULL`

Referenced by `_decode_rsa_pubkey()`, and `_get_rsa_dnskey()`.

5.412.2.240 `RSA>(* RSAPublicKey_dup_d)(RSA *rsa) = NULL`

Referenced by `_clone_dnskey_record_cb()`.

5.412.2.241 `int(* SHA1_Final_d)(unsigned char *md, SHA_CTX *c) = NULL`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.412.2.242 `int(* SHA1_Init_d)(SHA_CTX *c) = NULL`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.412.2.243 `int(* SHA1_Update_d)(SHA_CTX *c, const void *data, size_t len) = NULL`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.412.2.244 `int(* SHA256_Final_d)(unsigned char *md, SHA256_CTX *c) = NULL`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.412.2.245 `int(* SHA256_Init_d)(SHA256_CTX *c) = NULL`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.412.2.246 `int(* SHA512_Final_d)(unsigned char *md, SHA512_CTX *c) = NULL`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.412.2.247 `int(* SHA512_Init_d)(SHA512_CTX *c) = NULL`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.412.2.248 `int(* sk_num_d)(const _STACK *) = NULL`

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.412.2.249 `void*(* sk_pop_d)(_STACK *st) = NULL`

Referenced by `_dump_ocsp_response_cb()`.

5.412.2.250 `void(* sk_pop_free_d)(_STACK *st, void(*func)(void *)) = NULL`

Referenced by `ssl_stop()`.

5.412.2.251 `void*(* sk_value_d)(const _STACK *, int) = NULL`

Referenced by `_do_ocsp_validation()`.

5.412.2.252 `int(* SSL_accept_d)(SSL *ssl) = NULL`

Referenced by `tls_server_alloc()`.

5.412.2.253 `const char*(* SSL_CIPHER_get_name_d)(const SSL_CIPHER *cipher) = NULL`

Referenced by `tls_cipher()`, and `tls_suite()`.

5.412.2.254 `char*(* SSL_CIPHER_get_version_d)(const SSL_CIPHER *cipher) = NULL`

5.412.2.255 `int(* SSL_connect_d)(SSL *ssl) = NULL`

Referenced by `_ssl_connect_host()`, `_ssl_starttls()`, and `tls_client_alloc()`.

5.412.2.256 `long(* SSL_ctrl_d)(SSL *s, int cmd, long larg, void *parg) = NULL`

Referenced by `_ocsp_response_callback()`, and `_ssl_connect_host()`.

5.412.2.257 `long(* SSL_CTX_callback_ctrl_d)(SSL_CTX *, int, void(*) (void)) = NULL`

Referenced by `_ssl_connect_host()`.

5.412.2.258 `int(* SSL_CTX_check_private_key_d)(const SSL_CTX *ctx) = NULL`

Referenced by `tls_server_create()`.

5.412.2.259 `void(* SSL_CTX_free_d)(SSL_CTX *ctx) = NULL`

Referenced by `_ssl_shutdown()`, `ssl_verify_privkey()`, `tls_client_alloc()`, and `tls_server_destroy()`.

5.412.2.260 `SSL_CTX*(* SSL_CTX_new_d)(const SSL_METHOD *method) = NULL`

Referenced by `_ssl_get_client_context()`, `ssl_verify_privkey()`, `tls_client_alloc()`, and `tls_server_create()`.

5.412.2.261 `int(* SSL_CTX_set_cipher_list_d)(SSL_CTX *, const char *str) = NULL`

Referenced by `_ssl_get_client_context()`, and `tls_server_create()`.

5.412.2.262 `int(* SSL_do_handshake_d)(SSL *s) = NULL`

5.412.2.263 `void(* SSL_free_d)(SSL *ssl) = NULL`

Referenced by `_ssl_disconnect()`, `tls_client_alloc()`, `tls_free()`, and `tls_server_alloc()`.

5.412.2.264 `SSL_CIPHER*(* SSL_get_current_cipher_d)(const SSL *ssl) = NULL`

Referenced by `tls_bits()`, `tls_cipher()`, `tls_suite()`, and `tls_version()`.

5.412.2.265 `int(* SSL_get_error_d)(const SSL *s, int ret_code) = NULL`

Referenced by `tls_client_alloc()`, `tls_continue()`, `tls_error()`, and `tls_server_alloc()`.

5.412.2.266 `int(* SSL_get_fd_d)(const SSL *s) = NULL`

Referenced by `_ssl_disconnect()`.

5.412.2.267 `X509*(* SSL_get_peer_certificate_d)(const SSL *s) = NULL`

Referenced by `_do_ocsp_validation()`, and `_verify_dx_certificate()`.

5.412.2.268 `int(* SSL_get_read_ahead_d)(const SSL *s) = NULL`

5.412.2.269 `int(* SSL_get_rfd_d)(const SSL *s) = NULL`

Referenced by `_ssl_fd_loop()`.

5.412.2.270 `int(* SSL_get_shutdown_d)(const SSL *ssl) = NULL`

Referenced by `tls_status()`.

5.412.2.271 `const char*(* SSL_get_version_d)(const SSL *s) = NULL`

Referenced by `tls_cipher()`, and `tls_version()`.

5.412.2.272 `BIO*(* SSL_get_wbio_d)(const SSL *ssl) = NULL`

5.412.2.273 `int(* SSL_library_init_d)(void) = NULL`

Referenced by `_crypto_init()`, `_ssl_initialize()`, `encrypt_ctx_new()`, and `ssl_start()`.

5.412.2.274 `void(* SSL_load_error_strings_d)(void) = NULL`

Referenced by `_crypto_init()`, `_push_error_stack_openssl()`, `_ssl_initialize()`, `encrypt_ctx_new()`, and `ssl_start()`.

5.412.2.275 `SSL*(* SSL_new_d)(SSL_CTX *ctx) = NULL`

Referenced by `_ssl_connect_host()`, `_ssl_starttls()`, `tls_client_alloc()`, and `tls_server_alloc()`.

5.412.2.276 `int(* SSL_peek_d)(SSL *ssl, void *buf, int num) = NULL`

5.412.2.277 `int(* SSL_pending_d)(const SSL *ssl) = NULL`

5.412.2.278 `int(* SSL_read_d)(SSL *ssl, void *buf, int num) = NULL`

Referenced by `_sgnt_resolv_read_dmtpl_line()`, `_ssl_fd_loop()`, and `tls_read()`.

5.412.2.279 `void(* SSL_set_accept_state_d)(SSL *s) = NULL`

Referenced by `tls_server_alloc()`.

5.412.2.280 `void(* SSL_set_bio_d)(SSL *ssl, BIO *rbio, BIO *wbio) = NULL`

Referenced by `tls_client_alloc()`, and `tls_server_alloc()`.

5.412.2.281 `void(* SSL_set_connect_state_d)(SSL *s) = NULL`

Referenced by `tls_client_alloc()`.

5.412.2.282 `int(* SSL_set_fd_d)(SSL *s, int fd) = NULL`

Referenced by `_ssl_connect_host()`, and `_ssl_starttls()`.

5.412.2.283 `void(* SSL_set_read_ahead_d)(SSL *s, int yes) = NULL`

5.412.2.284 `int(* SSL_shutdown_d)(SSL *ssl) = NULL`

Referenced by `_ssl_disconnect()`, and `tls_free()`.

5.412.2.285 `char** SSL_version_str_d = NULL`

Definition at line 159 of file `symbols.c`.

Referenced by `lib_load_openssl()`.

5.412.2.286 `int(* SSL_want_d)(const SSL *s) = NULL`

5.412.2.287 `int(* SSL_write_d)(SSL *ssl, const void *buf, int num) = NULL`

Referenced by `_sgnt_resolv_dmtpl_issue_command()`, `_sgnt_resolv_dmtpl_write_data()`, `_ssl_fd_loop()`, and `tls_write()`.

5.412.2.288 `const char*(* SSLeay_version_d)(int t) = NULL`

5.412.2.289 `const SSL_METHOD*(* SSLv23_client_method_d)(void) = NULL`

Referenced by `_ssl_get_client_context()`, `ssl_verify_privkey()`, and `tls_client_alloc()`.

5.412.2.290 `const SSL_METHOD*(* SSLv23_server_method_d)(void) = NULL`

Referenced by `tls_server_create()`.

5.412.2.291 `const SSL_METHOD*(* TLSv1_server_method_d)(void) = NULL`

5.412.2.292 `int(* X509_check_issued_d)(X509 *issuer, X509 *subject) = NULL`

Referenced by `_do_ocsp_validation()`, and `_verify_dx_certificate()`.

5.412.2.293 `void(* X509_email_free_d)(struct stack_st_OPENSSL_STRING *sk) = NULL`

Referenced by `_do_ocsp_validation()`.

5.412.2.294 `struct stack_st_OPENSSL_STRING*(* X509_get1_ocsp_d)(X509 *x) = NULL`

Referenced by `_do_ocsp_validation()`.

5.412.2.295 `int(* X509_get_ext_count_d)(X509 *x) = NULL`

5.412.2.296 `X509_EXTENSION*(* X509_get_ext_d)(X509 *x, int loc) = NULL`

5.412.2.297 `X509_NAME*(* X509_get_subject_name_d)(X509 *a) = NULL`

Referenced by `_do_x509_validation()`, `_get_cert_subject_cn()`, and `_verify_certificate_callback()`.

5.412.2.298 `X509_LOOKUP_METHOD*(* X509_LOOKUP_file_d)(void) = NULL`

Referenced by `_do_x509_validation()`.

5.412.2.299 `char*(* X509_NAME_online_d)(X509_NAME *a, char *buf, int len) = NULL`

Referenced by `_do_x509_validation()`, and `_verify_certificate_callback()`.

5.412.2.300 `void(* X509_STORE_CTX_free_d)(X509_STORE_CTX *ctx) = NULL`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.412.2.301 `int(* X509_STORE_CTX_get_error_d)(X509_STORE_CTX *ctx) = NULL`

Referenced by `_validate_self_signed()`, and `_verify_certificate_callback()`.

5.412.2.302 `int(* X509_STORE_CTX_get_error_depth_d)(X509_STORE_CTX *ctx) = NULL`

Referenced by `_verify_certificate_callback()`.

5.412.2.303 `X509_STORE_CTX*(* X509_STORE_CTX_new_d)(void) = NULL`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.412.2.304 `void(* X509_STORE_free_d)(X509_STORE *v) = NULL`

Referenced by `_do_ocsp_validation()`, `_do_x509_validation()`, `_get_cert_store()`, and `_validate_self_signed()`.

5.412.2.305 `X509_STORE*(* X509_STORE_new_d)(void) = NULL`

Referenced by `_do_x509_validation()`, `_get_cert_store()`, and `_validate_self_signed()`.

5.412.2.306 `int(* X509_verify_cert_d)(X509_STORE_CTX *ctx) = NULL`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.412.2.307 `const char*(* X509_verify_cert_error_string_d)(long n) = NULL`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.413 src/providers/symbols.h File Reference

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#include <spf2/spf.h>
#include <spf2/spf_dns_zone.h>
#include <clamav.h>
#include <mysql/mysql.h>
#include <mysql/errmsg.h>
#include <openssl/conf.h>
#include <openssl/engine.h>
#include <openssl/err.h>
#include <openssl/rand.h>
#include <openssl/crypto.h>
#include <openssl/dh.h>
#include <openssl/ec.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/ssl.h>
#include <openssl/ocsp.h>
#include <lzo/lzodefs.h>
#include <lzo/lzoconf.h>
#include <lzo/lzoutil.h>
#include <lzo/lzo1x.h>
#include <libxml2/libxml/xmlmemory.h>
#include <libxml2/libxml/tree.h>
#include <libxml2/libxml/valid.h>
#include <libxml2/libxml/xpath.h>
#include <libxml2/libxml/xpathInternals.h>
#include <libxml2/libxml/parserInternals.h>
#include <libxml2/libxml/xmlerror.h>
#include <zlib.h>
#include <bzlib.h>
#include <tcutil.h>
#include <tcadb.h>
#include <tchdb.h>
#include <tcbdb.h>
```

```
#include <libmemcached/memcached.h>
#include <opendkim/dkim.h>
#include <dspam/libdspam.h>
#include <dspam/mysql_drv.h>
#include <jansson.h>
#include <gd.h>
#include <png.h>
#include <jpeglib.h>
#include <freetype2/ft2build.h>
#include <utf8proc.h>
```

Defines

- `#define lint`
- `#define CONFIG_DEFAULT ""`
- `#define LOGDIR "~/"`
- `#define M_BIND(x)`

Functions

- `STACK_OF(SSL_COMP)*(*SSL_COMP_get_compression_methods_d)(void)`

Variables

- `memcached_return_t(* memcached_flush_d)(memcached_st *ptr, time_t expiration)`
MEMCACHED.
- `void(* memcached_free_d)(memcached_st *ptr)`
- `const char *(* memcached_lib_version_d)(void)`
- `memcached_st *(* memcached_create_d)(memcached_st *ptr)`
- `const char *(* memcached_strerror_d)(const memcached_st *ptr, memcached_return_t rc)`
- `memcached_return_t(* memcached_behavior_set_d)(memcached_st *ptr, const memcached_behavior_t flag, uint64_t data)`
- `memcached_return_t(* memcached_delete_d)(memcached_st *ptr, const char *key, size_t key_length, time_t expiration)`
- `memcached_return_t(* memcached_server_add_with_weight_d)(memcached_st *ptr, const char *hostname, in_port_t port, uint32_t weight)`
- `memcached_return_t(* memcached_decrement_d)(memcached_st *ptr, const char *key, size_t key_length, uint32_t offset, uint64_t *value)`
- `memcached_return_t(* memcached_increment_d)(memcached_st *ptr, const char *key, size_t key_length, uint32_t offset, uint64_t *value)`
- `char *(* memcached_get_d)(memcached_st *ptr, const char *key, size_t key_length, size_t *value_length, uint32_t *flags, memcached_return_t *error)`
- `memcached_return_t(* memcached_add_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`
- `memcached_return_t(* memcached_set_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`
- `memcached_return_t(* memcached_append_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`
- `memcached_return_t(* memcached_prepend_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`

- `memcached_return_t(* memcached_replace_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`
- `memcached_return_t(* memcached_cas_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags, uint64_t cas)`
- `memcached_return_t(* memcached_decrement_with_initial_d)(memcached_st *ptr, const char *key, size_t key_length, uint64_t offset, uint64_t initial, time_t expiration, uint64_t *value)`
- `memcached_return_t(* memcached_increment_with_initial_d)(memcached_st *ptr, const char *key, size_t key_length, uint64_t offset, uint64_t initial, time_t expiration, uint64_t *value)`
- `const char *(* BZ2_bzlibVersion_d)(void)`

BZIP.

- `int(* BZ2_bzBuffToBuffDecompress_d)(char *dest, unsigned int *destLen, char *source, unsigned int sourceLen, int small, int verbosity)`
- `int(* BZ2_bzBuffToBuffCompress_d)(char *dest, unsigned int *destLen, char *source, unsigned int sourceLen, int blockSize100k, int verbosity, int workFactor)`
- `void(* cl_shutdown_d)(void)`

CLAMAV.

- `int(* lt_dlexit_d)(void)`
- `const char *(* cl_retver_d)(void)`
- `int(* cl_init_d)(unsigned int initoptions)`
- `const char *(* cl_strerror_d)(int clerror)`
- `struct cl_engine *(* cl_engine_new_d)(void)`
- `int(* cl_statfree_d)(struct cl_stat *dbstat)`
- `int(* cl_engine_free_d)(struct cl_engine *engine)`
- `int(* cl_engine_compile_d)(struct cl_engine *engine)`
- `int(* cl_statchkdir_d)(const struct cl_stat *dbstat)`
- `int(* cl_statinidir_d)(const char *dirname, struct cl_stat *dbstat)`
- `int(* cl_countsigs_d)(const char *path, unsigned int countoptions, unsigned int *sigs)`
- `int(* cl_engine_set_num_d)(struct cl_engine *engine, enum cl_engine_field field, long long num)`
- `int(* cl_engine_set_str_d)(struct cl_engine *engine, enum cl_engine_field field, const char *str)`
- `int(* cl_load_d)(const char *path, struct cl_engine *engine, unsigned int *signo, unsigned int dboptions)`
- `int(* cl_scandesc_d)(int desc, const char **virname, unsigned long int *scanned, const struct cl_engine *engine, unsigned int scanoptions)`
- `const char *(* dspam_version_d)(void)`

DSPAM.

- `int(* dspam_detach_d)(DSPAM_CTX *CTX)`
- `void(* dspam_destroy_d)(DSPAM_CTX *CTX)`
- `int(* dspam_init_driver_d)(DRIVER_CTX *DTX)`
- `int(* dspam_shutdown_driver_d)(DRIVER_CTX *DTX)`
- `int(* dspam_attach_d)(DSPAM_CTX *CTX, void *dbh)`
- `int(* dspam_process_d)(DSPAM_CTX *CTX, const char *message)`
- `DSPAM_CTX *(* dspam_create_d)(const char *username, const char *group, const char *home, int operating_mode, u_int32_t flags)`
- `DKIM_STAT(* dkim_eoh_d)(DKIM *dkim)`
- `void(* dkim_close_d)(DKIM_LIB *lib)`
- `uint32_t(* dkim_libversion_d)(void)`
- `DKIM_STAT(* dkim_free_d)(DKIM *dkim)`
- `char *(* dkim_geterror_d)(DKIM *dkim)`
- `DKIM_STAT(* dkim_eom_d)(DKIM *dkim, _Bool *testkey)`
- `const char *(* dkim_getresultstr_d)(DKIM_STAT result)`
- `DKIM_STAT(* dkim_body_d)(DKIM *dkim, u_char *buf, size_t len)`
- `DKIM_STAT(* dkim_header_d)(DKIM *dkim, u_char *hdr, size_t len)`
- `void(* dkim_mfree_d)(DKIM_LIB *libhandle, void *closure, void *ptr)`

- DKIM_STAT(* [dkim_chunk_d](#))(DKIM *dkim, unsigned char *chunkp, size_t len)
- DKIM_STAT(* [dkim_getsighdrx_d](#))(DKIM *dkim, u_char *buf, size_t len, size_t initial)
- int(* [dkim_test_dns_put_d](#))(DKIM *dkim, int class, int type, int prec, u_char *name, u_char *data)
- DKIM *(* [dkim_verify_d](#))(DKIM_LIB *libhandle, const unsigned char *id, void *memclosure, DKIM_STAT *statp)
- DKIM_LIB *(* [dkim_init_d](#))(void *(*mallocf)(void *closure, size_t nbytes), void(*freef)(void *closure, void *p))
- DKIM *(* [dkim_sign_d](#))(DKIM_LIB *libhandle, const unsigned char *id, void *memclosure, const dkim_sigkey_t secretkey, const unsigned char *selector, const unsigned char *domain, dkim_canon_t hdr_canon_alg, dkim_canon_t body_canon_alg, dkim_alg_t sign_alg, off_t length, DKIM_STAT *statp)
- FT_Error(* [FT_Done_FreeType_d](#))(FT_Library library)

FreeType.

- FT_Error(* [FT_Init_FreeType_d](#))(FT_Library *alibrary)
- void(* [FT_Library_Version_d](#))(FT_Library library, FT_Int *amajor, FT_Int *aminor, FT_Int *apatch)
- const char *(* [gdVersionString_d](#))(void)

GD.

- void(* [gdFree_d](#))(void *m)
- void *(* [gdImageGifPtr_d](#))(gdImagePtr im, int *size)
- void(* [gdImageDestroy_d](#))(gdImagePtr im)
- void *(* [gdImageJpegPtr_d](#))(gdImagePtr im, int *size, int quality)
- void(* [gdImageSetPixel_d](#))(gdImagePtr im, int x, int y, int color)
- gdImagePtr(* [gdImageCreate_d](#))(int sx, int sy)
- int(* [gdImageColorResolve_d](#))(gdImagePtr im, int r, int g, int b)
- char *(* [gdImageStringFT_d](#))(gdImage *im, int *brext, int fg, char *fontlist, double ptsize, double angle, int x, int y, char *string)
- const char *(* [jpeg_version_d](#))(void)

JPEG.

- const char *(* [lzo_version_string_d](#))(void)

LZO.

- int(* [__lzo_init_v2_d](#))(unsigned, int, int, int, int, int, int, int, int)
- lzo_uint32(* [lzo_adler32_d](#))(lzo_uint32 _adler, const lzo_bytpe _buf, lzo_uint _len)
- int(* [lzo1x_1_compress_d](#))(const lzo_byte *src, lzo_uint src_len, lzo_byte *dst, lzo_uintp dst_len, lzo_voidp wrkmem)
- int(* [lzo1x_decompress_safe_d](#))(const lzo_byte *src, lzo_uint src_len, lzo_byte *dst, lzo_uintp dst_len, lzo_voidp wrkmem)
- void(* [my_once_free_d](#))(void)

MYSQL.

- void(* [mysql_server_end_d](#))(void)
- void(* [mysql_thread_end_d](#))(void)
- int(* [mysql_ping_d](#))(MYSQL *mysql)
- void(* [mysql_close_d](#))(MYSQL *mysql)
- my_bool(* [mysql_thread_init_d](#))(void)
- const char *(* [mysql_get_server_info_d](#))(MYSQL *mysql)
- MYSQL *(* [mysql_init_d](#))(MYSQL *mysql)
- unsigned int(* [mysql_thread_safe_d](#))(void)
- int(* [mysql_stmt_fetch_d](#))(MYSQL_STMT *stmt)
- my_bool(* [mysql_stmt_close_d](#))(MYSQL_STMT *)
- unsigned int(* [mysql_errno_d](#))(MYSQL *mysql)
- const char *(* [mysql_error_d](#))(MYSQL *mysql)
- int(* [mysql_stmt_execute_d](#))(MYSQL_STMT *stmt)
- my_bool(* [mysql_embedded_d](#))(void)
- void(* [mysql_free_result_d](#))(MYSQL_RES *result)
- my_bool(* [mysql_stmt_reset_d](#))(MYSQL_STMT *stmt)

- `my_ulonglong(* mysql_insert_id_d)(MYSQL *mysql)`
- `unsigned long(* mysql_thread_id_d)(MYSQL *mysql)`
- `MYSQL_STMT *(* mysql_stmt_init_d)(MYSQL *mysql)`
- `MYSQL_ROW(* mysql_fetch_row_d)(MYSQL_RES *result)`
- `unsigned long(* mysql_get_client_version_d)(void)`
- `MYSQL_RES *(* mysql_store_result_d)(MYSQL *mysql)`
- `int(* mysql_stmt_store_result_d)(MYSQL_STMT *stmt)`
- `my_ulonglong(* mysql_affected_rows_d)(MYSQL *mysql)`
- `my_ulonglong(* mysql_num_rows_d)(MYSQL_RES *result)`
- `unsigned int(* mysql_stmt_errno_d)(MYSQL_STMT *stmt)`
- `const char *(* mysql_stmt_error_d)(MYSQL_STMT *stmt)`
- `my_bool(* mysql_stmt_free_result_d)(MYSQL_STMT *stmt)`
- `unsigned int(* mysql_num_fields_d)(MYSQL_RES *result)`
- `my_ulonglong(* mysql_stmt_num_rows_d)(MYSQL_STMT *stmt)`
- `MYSQL_FIELD *(* mysql_fetch_field_d)(MYSQL_RES *result)`
- `const char *(* mysql_character_set_name_d)(MYSQL *mysql)`
- `my_ulonglong(* mysql_stmt_insert_id_d)(MYSQL_STMT *stmt)`
- `my_ulonglong(* mysql_stmt_affected_rows_d)(MYSQL_STMT *stmt)`
- `MYSQL_RES *(* mysql_stmt_result_metadata_d)(MYSQL_STMT *stmt)`
- `int(* mysql_server_init_d)(int argc, char **argv, char **groups)`
- `int(* mysql_set_character_set_d)(MYSQL *mysql, const char *csname)`
- `my_bool(* mysql_stmt_bind_param_d)(MYSQL_STMT *stmt, MYSQL_BIND *bind)`
- `int(* mysql_options_d)(MYSQL *mysql, enum mysql_option option, const void *arg)`
- `int(* mysql_real_query_d)(MYSQL *mysql, const char *query, unsigned long length)`
- `int(* mysql_stmt_prepare_d)(MYSQL_STMT *stmt, const char *query, unsigned long length)`
- `unsigned long(* mysql_escape_string_d)(char *to, const char *from, unsigned long length)`
- `my_bool(* mysql_stmt_attr_set_d)(MYSQL_STMT *stmt, enum enum_stmt_attr_type attr_type, const void *attr)`
- `my_bool(* mysql_stmt_bind_result_d)(MYSQL_STMT *stmt, MYSQL_BIND *bind)`
- `MYSQL *(* mysql_real_connect_d)(MYSQL *mysql, const char *name, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag)`
- `DH *(* DH_new_d)(void)`

OPENSSL

- `char ** SSL_version_str_d`
- `RSA *(* RSA_new_d)(void)`
- `void(* DH_free_d)(DH *dh)`
- `int(* BIO_free_d)(BIO *a)`
- `int(* RAND_status_d)(void)`
- `void(* RSA_free_d)(RSA *r)`
- `void(* EVP_cleanup_d)(void)`
- `void(* OBJ_cleanup_d)(void)`
- `void(* BN_free_d)(BIGNUM *a)`
- `void(* RAND_cleanup_d)(void)`
- `void(* SSL_free_d)(SSL *ssl)`
- `int(* SSL_accept_d)(SSL *ssl)`
- `void *(* sk_pop_d)(_STACK *st)`
- `BN_CTX *(* BN_CTX_new_d)(void)`
- `int(* SSL_connect_d)(SSL *ssl)`
- `EC_KEY *(* EC_KEY_new_d)(void)`
- `void(* CRYPTO_free_d)(void *)`
- `void(* ENGINE_cleanup_d)(void)`
- `int(* SHA1_Init_d)(SHA_CTX *c)`
- `void(* BIO_free_all_d)(BIO *a)`

- [int\(* CRYPTO_num_locks_d\)\(void\)](#)
- [int\(* SSL_library_init_d\)\(void\)](#)
- [int\(* SSL_want_d\)\(const SSL *s\)](#)
- [int\(* SSL_shutdown_d\)\(SSL *ssl\)](#)
- [void\(* ERR_clear_error_d\)\(void\)](#)
- [int\(* sk_num_d\)\(const _STACK *\)](#)
- [void\(* BIO_sock_cleanup_d\)\(void\)](#)
- [void\(* ERR_free_strings_d\)\(void\)](#)
- [SSL *\(* SSL_new_d\)\(SSL_CTX *ctx\)](#)
- [const EVP_MD *\(* EVP_md4_d\)\(void\)](#)
- [const EVP_MD *\(* EVP_md5_d\)\(void\)](#)
- [const EVP_MD *\(* EVP_sha_d\)\(void\)](#)
- [void\(* COMP_zlib_cleanup_d\)\(void\)](#)
- [int\(* SSL_get_fd_d\)\(const SSL *s\)](#)
- [int\(* SSL_do_handshake_d\)\(SSL *s\)](#)
- [void\(* BN_CTX_free_d\)\(BN_CTX *ctx\)](#)
- [const EVP_MD *\(* EVP_sha1_d\)\(void\)](#)
- [void\(* EC_KEY_free_d\)\(EC_KEY *key\)](#)
- [int\(* SSL_get_rfd_d\)\(const SSL *s\)](#)
- [EVP_PKEY *\(* EVP_PKEY_new_d\)\(void\)](#)
- [void\(* BN_CTX_start_d\)\(BN_CTX *ctx\)](#)
- [const char *\(* OBJ_nid2sn_d\)\(int n\)](#)
- [int\(* SHA256_Init_d\)\(SHA256_CTX *c\)](#)
- [int\(* SHA512_Init_d\)\(SHA512_CTX *c\)](#)
- [int\(* SSL_set_fd_d\)\(SSL *s, int fd\)](#)
- [const EVP_MD *\(* EVP_sha224_d\)\(void\)](#)
- [const EVP_MD *\(* EVP_sha256_d\)\(void\)](#)
- [const EVP_MD *\(* EVP_sha384_d\)\(void\)](#)
- [const EVP_MD *\(* EVP_sha512_d\)\(void\)](#)
- [void\(* OBJ_NAME_cleanup_d\)\(int type\)](#)
- [void\(* SSL_CTX_free_d\)\(SSL_CTX *ctx\)](#)
- [int\(* SSL_pending_d\)\(const SSL *ssl\)](#)
- [int\(* BN_num_bits_d\)\(const BIGNUM *\)](#)
- [int\(* X509_get_ext_count_d\)\(X509 *x\)](#)
- [RSA *\(* RSAPublicKey_dup_d\)\(RSA *rsa\)](#)
- [char *\(* BN_bn2dec_d\)\(const BIGNUM *a\)](#)
- [char *\(* BN_bn2hex_d\)\(const BIGNUM *a\)](#)
- [int\(* EVP_MD_size_d\)\(const EVP_MD *md\)](#)
- [unsigned long\(* ERR_get_error_d\)\(void\)](#)
- [void\(* CONF_modules_unload_d\)\(int all\)](#)
- [void\(* HMAC_CTX_init_d\)\(HMAC_CTX *ctx\)](#)
- [void\(* SSL_load_error_strings_d\)\(void\)](#)
- [int\(* EVP_MD_type_d\)\(const EVP_MD *md\)](#)
- [void\(* ECDSA_SIG_free_d\)\(ECDSA_SIG *a\)](#)
- [X509_STORE *\(* X509_STORE_new_d\)\(void\)](#)
- [void\(* SSL_set_accept_state_d\)\(SSL *s\)](#)
- [void\(* SSL_set_connect_state_d\)\(SSL *s\)](#)
- [unsigned long\(* ERR_peek_error_d\)\(void\)](#)
- [const EVP_MD *\(* EVP_ripemd160_d\)\(void\)](#)
- [const char *\(* SSLeay_version_d\)\(int t\)](#)
- [void\(* ERR_load_crypto_strings_d\)\(void\)](#)
- [void\(* ERR_print_errors_fp_d\)\(FILE *fp\)](#)
- [BIO *\(* SSL_get_wbio_d\)\(const SSL *ssl\)](#)

- void(* [EC_GROUP_free_d](#))(EC_GROUP *group)
- void(* [EC_POINT_free_d](#))(EC_POINT *point)
- void(* [X509_STORE_free_d](#))(X509_STORE *v)
- int(* [SSL_get_read_ahead_d](#))(const SSL *s)
- int(* [DH_check_d](#))(const DH *dh, int *ret)
- int(* [EC_KEY_generate_key_d](#))(EC_KEY *key)
- void(* [ASN1_STRING_TABLE_cleanup_d](#))(void)
- void(* [HMAC_CTX_cleanup_d](#))(HMAC_CTX *ctx)
- int(* [SSL_get_shutdown_d](#))(const SSL *ssl)
- void(* [sk_value_d](#))(const _STACK *, int)
- void(* [CRYPTO_cleanup_all_ex_data_d](#))(void)
- void(* [EVP_MD_CTX_init_d](#))(EVP_MD_CTX *ctx)
- OCSF_REQUEST *(* [OCSF_REQUEST_new_d](#))(void)
- int(* [EC_KEY_check_key_d](#))(const EC_KEY *key)
- int(* [EVP_MD_CTX_cleanup_d](#))(EVP_MD_CTX *ctx)
- void(* [OCSF_REQUEST_free_d](#))(OCSF_REQUEST *a)
- const EVP_CIPHER *(* [EVP_aes_256_gcm_d](#))(void)
- const EVP_CIPHER *(* [EVP_aes_256_cbc_d](#))(void)
- int(* [SSL_peek_d](#))(SSL *ssl, void *buf, int num)
- void(* [SSL_set_read_ahead_d](#))(SSL *s, int yes)
- EVP_CIPHER_CTX *(* [EVP_CIPHER_CTX_new_d](#))(void)
- int(* [OCSF_check_nonce_d](#))(void *req, void *bs)
- int(* [X509_verify_cert_d](#))(X509_STORE_CTX *ctx)
- void(* [EC_GROUP_clear_free_d](#))(EC_GROUP *group)
- void(* [OCSF_RESPONSE_free_d](#))(OCSF_RESPONSE *a)
- X509_STORE_CTX *(* [X509_STORE_CTX_new_d](#))(void)
- X509_NAME *(* [X509_get_subject_name_d](#))(X509 *a)
- EC_KEY *(* [EC_KEY_new_by_curve_name_d](#))(int nid)
- int(* [BN_hex2bn_d](#))(BIGNUM **a, const char *str)
- int(* [SSL_read_d](#))(SSL *ssl, void *buf, int num)
- int(* [i2d_X509_d](#))(X509 *a, unsigned char **out)
- const char *(* [SSL_get_version_d](#))(const SSL *s)
- int(* [RAND_bytes_d](#))(unsigned char *buf, int num)
- void(* [EVP_CIPHER_CTX_init_d](#))(EVP_CIPHER_CTX *a)
- void(* [OCSF_BASICRESP_free_d](#))(OCSF_BASICRESP *a)
- void(* [EVP_CIPHER_CTX_free_d](#))(EVP_CIPHER_CTX *a)
- X509_LOOKUP_METHOD *(* [X509_LOOKUP_file_d](#))(void)
- int(* [BN_cmp_d](#))(const BIGNUM *a, const BIGNUM *b)
- const SSL_METHOD *(* [TLSv1_server_method_d](#))(void)
- int(* [EVP_CIPHER_nid_d](#))(const EVP_CIPHER *cipher)
- void(* [OPENSSL_add_all_algorithms_noconf_d](#))(void)
- int(* [SSL_get_error_d](#))(const SSL *s, int ret_code)
- const SSL_METHOD *(* [SSLv23_client_method_d](#))(void)
- const SSL_METHOD *(* [SSLv23_server_method_d](#))(void)
- X509 *(* [SSL_get_peer_certificate_d](#))(const SSL *s)
- int(* [EVP_CIPHER_CTX_cleanup_d](#))(EVP_CIPHER_CTX *a)
- const char *(* [OCSF_response_status_str_d](#))(long s)
- int(* [OCSF_response_status_d](#))(OCSF_RESPONSE *resp)
- int(* [SHA1_Final_d](#))(unsigned char *md, SHA_CTX *c)
- void(* [X509_STORE_CTX_free_d](#))(X509_STORE_CTX *ctx)
- BIO *(* [BIO_new_socket_d](#))(int sock, int close_flag)
- EC_GROUP *(* [EC_GROUP_new_by_curve_name_d](#))(int nid)
- EC_POINT *(* [EC_POINT_new_d](#))(const EC_GROUP *group)

- int(* [BN_bn2bin_d](#))(const BIGNUM *, unsigned char *)
- BIO>(* [BIO_new_fp_d](#))(FILE *stream, int close_flag)
- X509_EXTENSION(*([X509_get_ext_d](#))(X509 *x, int loc)
- SSL_CTX(*([SSL_CTX_new_d](#))(const SSL_METHOD *method)
- void(* [SSL_set_bio_d](#))(SSL *ssl, BIO *rbio, BIO *wbio)
- unsigned char(*([ASN1_STRING_data_d](#))(ASN1_STRING *x)
- int(* [BN_bn2mpi_d](#))(const BIGNUM *a, unsigned char *to)
- int(* [SSL_CTX_check_private_key_d](#))(const SSL_CTX *ctx)
- int(* [SSL_write_d](#))(SSL *ssl, const void *buf, int num)
- void(* [sk_pop_free_d](#))(_STACK *st, void(*func)(void *))
- int(* [X509_STORE_CTX_get_error_d](#))(X509_STORE_CTX *ctx)
- void(*([OCSP_request_add0_id_d](#))(void *req, void *cid)
- int(* [EVP_CIPHER_iv_length_d](#))(const EVP_CIPHER *cipher)
- const char(*([X509_verify_cert_error_string_d](#))(long n)
- int(* [SHA256_Final_d](#))(unsigned char *md, SHA256_CTX *c)
- int(* [SHA512_Final_d](#))(unsigned char *md, SHA512_CTX *c)
- int(* [X509_check_issued_d](#))(X509 *issuer, X509 *subject)
- SSL_CIPHER(*([SSL_get_current_cipher_d](#))(const SSL *ssl)
- int(* [EVP_CIPHER_block_size_d](#))(const EVP_CIPHER *cipher)
- int(* [EVP_CIPHER_key_length_d](#))(const EVP_CIPHER *cipher)
- void(*([OCSP_response_get1_basic_d](#))(OCSP_RESPONSE *resp)
- const EC_GROUP(*([EC_KEY_get0_group_d](#))(const EC_KEY *key)
- const EVP_MD(*([EVP_get_digestbyname_d](#))(const char *name)
- long(* [SSL_ctrl_d](#))(SSL *s, int cmd, long larg, void *parg)
- int(* [i2o_ECPublicKey_d](#))(EC_KEY *key, unsigned char **out)
- int(* [SSL_CTX_set_cipher_list_d](#))(SSL_CTX *, const char *str)
- int(* [i2d_ECPrivateKey_d](#))(EC_KEY *key, unsigned char **out)
- char(*([SSL_CIPHER_get_version_d](#))(const SSL_CIPHER *cipher)
- int(* [EVP_CIPHER_CTX_iv_length_d](#))(const EVP_CIPHER_CTX *ctx)
- int(* [EVP_DigestInit_d](#))(EVP_MD_CTX *ctx, const EVP_MD *type)
- int(* [X509_STORE_CTX_get_error_depth_d](#))(X509_STORE_CTX *ctx)
- int(* [EC_KEY_set_group_d](#))(EC_KEY *key, const EC_GROUP *group)
- int(* [EVP_CIPHER_CTX_block_size_d](#))(const EVP_CIPHER_CTX *ctx)
- int(* [EVP_CIPHER_CTX_key_length_d](#))(const EVP_CIPHER_CTX *ctx)
- int(* [RAND_load_file_d](#))(const char *filename, long max_bytes)
- void(* [ERR_remove_thread_state_d](#))(const CRYPTO_THREADID *tid)
- unsigned long(* [EVP_CIPHER_flags_d](#))(const EVP_CIPHER *cipher)
- int(* [i2d_OCSP_CERTID_d](#))(OCSP_CERTID *a, unsigned char **out)
- int(* [OCSP_REQ_CTX_set1_req_d](#))(OCSP_REQ_CTX *rctx, void *req)
- struct stack_st_OPENSSL_STRING(*([X509_get1_ocsp_d](#))(X509 *x)
- void(* [X509_email_free_d](#))(struct stack_st_OPENSSL_STRING *sk)
- const BIGNUM(*([EC_KEY_get0_private_key_d](#))(const EC_KEY *key)
- const EVP_CIPHER(*([EVP_get_cipherbyname_d](#))(const char *name)
- int(* [EVP_PKEY_set1_RSA_d](#))(EVP_PKEY *pkey, struct rsa_st *key)
- int(* [SHA1_Update_d](#))(SHA_CTX *c, const void *data, size_t len)
- const char(*([SSL_CIPHER_get_name_d](#))(const SSL_CIPHER *cipher)
- BIGNUM(*([BN_mpi2bn_d](#))(unsigned char *s, int len, BIGNUM *ret)
- const EC_POINT(*([EC_KEY_get0_public_key_d](#))(const EC_KEY *key)
- int(* [EC_GROUP_precompute_mult_d](#))(EC_GROUP *group, BN_CTX *ctx)
- int(* [EC_KEY_set_private_key_d](#))(EC_KEY *key, const BIGNUM *prv)
- int(* [EVP_CIPHER_CTX_set_padding_d](#))(EVP_CIPHER_CTX *c, int pad)
- long(* [SSL_CTX_callback_ctrl_d](#))(SSL_CTX *, int, void(*) (void))
- char(*([X509_NAME_online_d](#))(X509_NAME *a, char *buf, int len)

- `size_t(* BUF_strlcat_d)(char *dst, const char *src, size_t siz)`
- `int(* BIO_vprintf_d)(BIO *bio, const char *format, va_list args)`
- `int(* EC_KEY_set_public_key_d)(EC_KEY *key, const EC_POINT *pub)`
- `ASN1_STRING *(* X509_NAME_ENTRY_get_data_d)(X509_NAME_ENTRY *ne)`
- `int(* i2d_ECDSA_SIG_d)(const ECDSA_SIG *sig, unsigned char **pp)`
- `X509 *(* X509_STORE_CTX_get_current_cert_d)(X509_STORE_CTX *ctx)`
- `int(* SSL_CIPHER_get_bits_d)(const SSL_CIPHER *c, int *alg_bits)`
- `int(* i2d_OCSP_RESPONSE_d)(OCSP_RESPONSE *a, unsigned char **out)`
- `struct stack_st_X509 *(* SSL_get_peer_cert_chain_d)(const SSL *s)`
- `unsigned long(* EVP_CIPHER_CTX_flags_d)(const EVP_CIPHER_CTX *ctx)`
- `int(* OCSP_REQUEST_print_d)(BIO *bp, void *a, unsigned long flags)`
- `void(* CRYPTO_set_id_callback_d)(unsigned long(*id_function)(void))`
- `int(* SHA256_Update_d)(SHA256_CTX *c, const void *data, size_t len)`
- `int(* SHA512_Update_d)(SHA512_CTX *c, const void *data, size_t len)`
- `int(* X509_STORE_set_flags_d)(X509_STORE *ctx, unsigned long flags)`
- `point_conversion_form_t(* EC_KEY_get_conv_form_d)(const EC_KEY *key)`
- `long(* SSL_CTX_ctrl_d)(SSL_CTX *ctx, int cmd, long larg, void *parg)`
- `void(* ERR_error_string_n_d)(unsigned long e, char *buf, size_t len)`
- `BIGNUM *(* ASN1_INTEGER_to_BN_d)(const ASN1_INTEGER *ai, BIGNUM *bn)`
- `X509_NAME_ENTRY *(* X509_NAME_get_entry_d)(X509_NAME *name, int loc)`
- `BIGNUM *(* BN_bin2bn_d)(const unsigned char *s, int len, BIGNUM *ret)`
- `int(* EVP_DigestUpdate_d)(EVP_MD_CTX *ctx, const void *d, size_t cnt)`
- `int(* OCSP_sendreq_nbio_d)(OCSP_RESPONSE **presp, OCSP_REQ_CTX *rctx)`
- `int(* HMAC_Final_d)(HMAC_CTX *ctx, unsigned char *md, unsigned int *len)`
- `int(* OCSP_request_add1_nonce_d)(void *req, unsigned char *val, int len)`
- `int(* HMAC_Update_d)(HMAC_CTX *ctx, const unsigned char *data, size_t len)`
- `int(* X509_NAME_get_index_by_NID_d)(X509_NAME *name, int nid, int lastpos)`
- `int(* SSL_CTX_use_certificate_chain_file_d)(SSL_CTX *ctx, const char *file)`
- `int(* ASN1_GENERALIZEDTIME_print_d)(BIO *fp, const ASN1_GENERALIZEDTIME *a)`
- `void(* EC_KEY_set_conv_form_d)(EC_KEY *eckey, point_conversion_form_t cform)`
- `int(* OCSP_RESPONSE_print_d)(BIO *bp, OCSP_RESPONSE *o, unsigned long flags)`
- `void *(* OCSP_cert_to_id_d)(const EVP_MD *dgst, X509 *subject, X509 *issuer)`
- `int(* EVP_DigestFinal_d)(EVP_MD_CTX *ctx, unsigned char *md, unsigned int *s)`
- `int(* EVP_DigestInit_ex_d)(EVP_MD_CTX *ctx, const EVP_MD *type, ENGINE *impl)`
- `int(* SSL_CTX_use_PrivateKey_file_d)(SSL_CTX *ctx, const char *file, int type)`
- `X509_LOOKUP *(* X509_STORE_add_lookup_d)(X509_STORE *v, X509_LOOKUP_METHOD *m)`
- `int(* EVP_CIPHER_CTX_ctrl_d)(EVP_CIPHER_CTX *ctx, int type, int arg, void *ptr)`
- `int(* X509_NAME_get_text_by_NID_d)(X509_NAME *name, int nid, char *buf, int len)`
- `EC_KEY *(* o2i_ECPrivateKey_d)(EC_KEY **key, const unsigned char **in, long len)`
- `int(* EVP_DigestFinal_ex_d)(EVP_MD_CTX *ctx, unsigned char *md, unsigned int *s)`
- `int(* EVP_EncryptFinal_ex_d)(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)`
- `EC_KEY *(* d2i_ECPrivateKey_d)(EC_KEY **key, const unsigned char **in, long len)`
- `unsigned char *(* SHA512_d)(const unsigned char *d, size_t n, unsigned char *md)`
- `int(* EVP_DecryptFinal_ex_d)(EVP_CIPHER_CTX *ctx, unsigned char *outm, int *outl)`
- `void(* EC_GROUP_set_point_conversion_form_d)(EC_GROUP *, point_conversion_form_t)`
- `void(* ED25519_keypair_d)(uint8_t out_public_key[32], uint8_t out_private_key[64])`
- `void(* ERR_put_error_d)(int lib, int func, int reason, const char *file, int line)`
- `ECDSA_SIG *(* d2i_ECDSA_SIG_d)(ECDSA_SIG **sig, const unsigned char **pp, long len)`
- `int(* DH_generate_parameters_ex_d)(DH *dh, int prime_len, int generator, BN_GENCB *cb)`
- `ECDSA_SIG *(* ECDSA_do_sign_d)(const unsigned char *dgst, int dgst_len, EC_KEY *eckey)`
- `int(* X509_STORE_load_locations_d)(X509_STORE *ctx, const char *file, const char *path)`
- `OCSP_REQ_CTX *(* OCSP_sendreq_new_d)(BIO *io, const char *path, void *req, int maxline)`
- `void(* SSL_CTX_set_verify_d)(SSL_CTX *ctx, int mode, int(*cb)(int, X509_STORE_CTX *))`

- `EC_POINT *(* EC_POINT_hex2point_d)(const EC_GROUP *, const char *, EC_POINT *, BN_CTX *)`
- `int(* CRYPTO_set_locked_mem_functions_d)(void (*)(m)(size_t), void(*free_func)(void *))`
- `int(* OCSP_REQ_CTX_add1_header_d)(OCSP_REQ_CTX *rctx, const char *name, const char *value)`
- `void(* X509_STORE_CTX_set_chain_d)(struct x509_store_ctx_st *ctx, struct stack_st_X509 *sk)`
- `int(* SSL_CTX_load_verify_locations_d)(SSL_CTX *ctx, const char *CAfile, const char *CApath)`
- `OCSP_RESPONSE *(* d2i_OCSP_RESPONSE_d)(OCSP_RESPONSE **a, const unsigned char **in, long len)`
- `int(* OCSP_parse_url_d)(const char *url, char **phost, char **pport, char **ppath, int *pssl)`
- `int(* HMAC_Init_ex_d)(HMAC_CTX *ctx, const void *key, int len, const EVP_MD *md, ENGINE *impl)`
- `int(* EC_POINT_cmp_d)(const EC_GROUP *group, const EC_POINT *a, const EC_POINT *b, BN_CTX *ctx)`
- `void(* SSL_CTX_set_tmp_dh_callback_d)(SSL_CTX *ctx, DH *(*dh)(SSL *ssl, int is_export, int keylength))`
- `int(* ECDSA_do_verify_d)(const unsigned char *dgst, int dgst_len, const ECDSA_SIG *sig, EC_KEY *eckey)`
- `int(* X509_check_host_d)(X509 *x, const char *chk, size_t chklen, unsigned int flags, char **peername)`
- `int(* CRYPTO_set_mem_functions_d)(void (*)(m)(size_t), void (*)(r)(void *, size_t), void (*f)(void *))`
- `int(* X509_STORE_CTX_init_d)(X509_STORE_CTX *ctx, X509_STORE *store, X509 *x509, STACK_OF(X509) *chain)`
- `unsigned long(* ERR_peek_error_line_data_d)(const char **file, int *line, const char **data, int *flags)`
- `char *(* EC_POINT_point2hex_d)(const EC_GROUP *, const EC_POINT *, point_conversion_form_t form, BN_CTX *)`
- `void(* CRYPTO_set_locking_callback_d)(void(*locking_function)(int mode, int n, const char *file, int line))`
- `int(* EVP_VerifyFinal_d)(EVP_MD_CTX *ctx, const unsigned char *sigbuf, unsigned int siglen, EVP_PKEY *pkey)`
- `void(* SSL_CTX_set_tmp_ecdh_callback_d)(SSL_CTX *ctx, EC_KEY *(*ecdh)(SSL *ssl, int is_export, int keylength))`
- `int(* EVP_DecryptUpdate_d)(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl, const unsigned char *in, int inl)`
- `int(* EVP_EncryptUpdate_d)(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl, const unsigned char *in, int inl)`
- `int(* OCSP_basic_verify_d)(void *bs, struct stack_st_X509 *certs, struct x509_store_st *st, unsigned long flags)`
- `int(* OCSP_check_validity_d)(ASN1_GENERALIZEDTIME *thisupd, ASN1_GENERALIZEDTIME *nextupd, long sec, long maxsec)`
- `int(* ED25519_sign_d)(uint8_t *out_sig, const uint8_t *message, size_t message_len, const uint8_t private_key[64])`
- `int(* EC_POINT_oct2point_d)(const EC_GROUP *group, EC_POINT *p, const unsigned char *buf, size_t len, BN_CTX *ctx)`
- `void(* ED25519_keypair_from_seed_d)(uint8_t out_public_key[32], uint8_t out_private_key[64], const uint8_t seed[32])`
- `int(* EVP_Digest_d)(const void *data, size_t count, unsigned char *md, unsigned int *size, const EVP_MD *type, ENGINE *impl)`
- `int(* ED25519_verify_d)(const uint8_t *message, size_t message_len, const uint8_t signature[64], const uint8_t public_key[32])`
- `int(* EVP_DecryptInit_ex_d)(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *impl, const unsigned char *key, const unsigned char *iv)`
- `int(* EVP_EncryptInit_ex_d)(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *impl, const unsigned char *key, const unsigned char *iv)`
- `int(* EC_POINT_mul_d)(const EC_GROUP *group, EC_POINT *r, const BIGNUM *g_scalar, const EC_POINT *point, const BIGNUM *p_scalar, BN_CTX *ctx)`
- `size_t(* EC_POINT_point2oct_d)(const EC_GROUP *group, const EC_POINT *p, point_conversion_form_t form, unsigned char *buf, size_t len, BN_CTX *ctx)`
- `int(* ECDH_compute_key_d)(void *out, size_t outlen, const EC_POINT *pub_key, EC_KEY *ecdh, void *(*KDF)(const void *in, size_t inlen, void *out, size_t *outlen))`
- `int(* OCSP_resp_find_status_d)(void *bs, void *id, int *status, int *reason, ASN1_GENERALIZEDTIME **revtime, ASN1_GENERALIZEDTIME **thisupd, ASN1_GENERALIZEDTIME **nextupd)`
- `png_uint_32(* png_access_version_number_d)(void)`
PNG.
- `void(* SPF_server_free_d)(SPF_server_t *sp)`
SPF.
- `void(* SPF_request_free_d)(SPF_request_t *sr)`
- `void(* SPF_response_free_d)(SPF_response_t *rp)`
- `const char *(* SPF_strerror_d)(SPF_reason_t reason)`
- `const char *(* SPF_strerror_d)(SPF_result_t result)`
- `const char *(* SPF_strerror_d)(SPF_errcode_t spf_err)`
- `SPF_reason_t(* SPF_response_reason_d)(SPF_response_t *rp)`
- `SPF_result_t(* SPF_response_result_d)(SPF_response_t *rp)`

- `SPF_request_t` `*(SPF_request_new_d)(SPF_server_t *spf_server)`
- `void` `*(SPF_get_lib_version_d)(int *major, int *minor, int *patch)`
- `int` `*(SPF_request_set_env_from_d)(SPF_request_t *sr, const char *from)`
- `SPF_server_t` `*(SPF_server_new_d)(SPF_server_dnstype_t dnstype, int debug)`
- `SPF_errcode_t` `*(SPF_request_set_helo_dom_d)(SPF_request_t *sr, const char *dom)`
- `SPF_errcode_t` `*(SPF_request_set_ipv4_d)(SPF_request_t *sr, struct in_addr addr)`
- `SPF_errcode_t` `*(SPF_request_set_ipv6_d)(SPF_request_t *sr, struct in6_addr addr)`
- `SPF_dns_server_t` `*(SPF_dns_zone_new_d)(SPF_dns_server_t *layer_below, const char *name, int debug)`
- `SPF_errcode_t` `*(SPF_request_query_mailfrom_d)(SPF_request_t *spf_request, SPF_response_t **spf_responsep)`
- `SPF_errcode_t` `*(SPF_dns_zone_add_str_d)(SPF_dns_server_t *spf_dns_server, const char *domain, ns_type rr_type, SPF_dns_stat_t hermo, const char *data)`
- `char **` `tcversion_d`

TOKYO.

- `TCHDB` `*(tchdbnew_d)(void)`
- `void` `*(tcfree_d)(void *ptr)`
- `void` `*(tchdbdel_d)(TCHDB *hdb)`
- `bool` `*(tchdbsync_d)(TCHDB *hdb)`
- `int` `*(tchdbecode_d)(TCHDB *hdb)`
- `void` `*(tcndbdel_d)(TCNDB *tree)`
- `bool` `*(tchdbclose_d)(TCHDB *hdb)`
- `void` `*(tclistdel_d)(TCLIST *list)`
- `TCNDB` `*(tcndbdup_d)(TCNDB *ndb)`
- `void` `*(tctreeclear_d)(TCTREE *tree)`
- `bool` `*(tchdbsetmutex_d)(TCHDB *hdb)`
- `uint64_t` `*(tchdbfsiz_d)(TCHDB *hdb)`
- `uint64_t` `*(tchdbnum_d)(TCHDB *hdb)`
- `uint64_t` `*(tcndbnum_d)(TCNDB *ndb)`
- `void` `*(tcndbiterinit_d)(TCNDB *ndb)`
- `char` `*(tcndbiternext2_d)(TCNDB *ndb)`
- `int` `*(tclistnum_d)(const TCLIST *list)`
- `const char` `*(tchdberrmsg_d)(int ecodes)`
- `const char` `*(tchdbpath_d)(TCHDB *hdb)`
- `TCLIST` `*(tctreekeys_d)(const TCTREE *tree)`
- `TCLIST` `*(tctreevals_d)(const TCTREE *tree)`
- `TCNDB` `*(tcndbnew2_d)(TCCMP cmp, void *cmpop)`
- `bool` `*(tchdbdefrag_d)(TCHDB *hdb, int64_t step)`
- `bool` `*(tchdbsetdfunit_d)(TCHDB *hdb, int32_t dfunit)`
- `bool` `*(tchdbout_d)(TCHDB *hdb, const void *kbuf, int ksiz)`
- `bool` `*(tcndbout_d)(TCNDB *ndb, const void *kbuf, int ksiz)`
- `bool` `*(tchdbopen_d)(TCHDB *hdb, const char *path, int omode)`
- `const void` `*(tclistval_d)(const TCLIST *list, int index, int *sp)`
- `void` `*(tchdbget_d)(TCHDB *hdb, const void *kbuf, int ksiz, int *sp)`
- `void` `*(tcndbget3_d)(TCNDB *ndb, const void *kbuf, int ksiz, int *sp)`
- `void` `*(tcndbget_d)(TCNDB *ndb, const void *kbuf, int ksiz, int *sp)`
- `TCLIST` `*(tcndbfwmkeys_d)(TCNDB *ndb, const void *pbuf, int psiz, int max)`
- `bool` `*(tchdbtune_d)(TCHDB *hdb, int64_t bnum, int8_t apow, int8_t fpow, uint8_t opts)`
- `bool` `*(tchdboptimize_d)(TCHDB *hdb, int64_t bnum, int8_t apow, int8_t fpow, uint8_t opts)`
- `bool` `*(tcndbputkeep_d)(TCNDB *ndb, const void *kbuf, int ksiz, const void *vbuf, int vsiz)`
- `bool` `*(tchdbputasyn_d)(TCHDB *hdb, const void *kbuf, int ksiz, const void *vbuf, int vsiz)`
- `bool` `*(tcndbgetboth_d)(TCNDB *ndb, const void *kbuf, int ksiz, void **rkbuf, int *rksiz, void **rvbuf, int *rvsiz)`
- `const char` `*(jansson_version_d)(void)`

Jansson.

- `int(* json_array_append_d)(json_t *array, json_t *value)`
- `int(* json_array_insert_d)(json_t *array, size_t index, json_t *value)`
- `int(* json_array_set_d)(json_t *array, size_t index, json_t *value)`
- `json_t *(* json_array_d)(void)`
- `int(* json_array_append_new_d)(json_t *array, json_t *value)`
- `int(* json_array_clear_d)(json_t *array)`
- `int(* json_array_extend_d)(json_t *array, json_t *other)`
- `json_t *(* json_array_get_d)(const json_t *array, size_t index)`
- `int(* json_array_insert_new_d)(json_t *array, size_t index, json_t *value)`
- `int(* json_array_remove_d)(json_t *array, size_t index)`
- `int(* json_array_set_new_d)(json_t *array, size_t index, json_t *value)`
- `size_t(* json_array_size_d)(const json_t *array)`
- `json_t *(* json_copy_d)(json_t *value)`
- `void(* json_decref_d)(json_t *json)`
- `json_t *(* json_deep_copy_d)(json_t *value)`
- `void(* json_delete_d)(json_t *json)`
- `int(* json_dump_file_d)(const json_t *json, const char *path, size_t flags)`
- `int(* json_dumpf_d)(const json_t *json, FILE *output, size_t flags)`
- `char *(* json_dumps_d)(const json_t *json, size_t flags)`
- `int(* json_equal_d)(json_t *value1, json_t *value2)`
- `json_t *(* json_false_d)(void)`
- `const char *(* json_type_string_d)(json_t *json)`
- `json_t *(* json_incref_d)(json_t *json)`
- `json_t *(* json_integer_d)(json_int_t value)`
- `int(* json_integer_set_d)(json_t *integer, json_int_t value)`
- `json_int_t(* json_integer_value_d)(const json_t *integer)`
- `json_t *(* json_load_file_d)(const char *path, size_t flags, json_error_t *error)`
- `json_t *(* json_loadf_d)(FILE *input, size_t flags, json_error_t *error)`
- `json_t *(* json_loads_d)(const char *input, size_t flags, json_error_t *error)`
- `json_t *(* json_null_d)(void)`
- `double(* json_number_value_d)(const json_t *json)`
- `json_t *(* json_object_d)(void)`
- `int(* json_object_clear_d)(json_t *object)`
- `int(* json_object_del_d)(json_t *object, const char *key)`
- `json_t *(* json_object_get_d)(const json_t *object, const char *key)`
- `void *(* json_object_iter_d)(json_t *object)`
- `void *(* json_object_iter_at_d)(json_t *object, const char *key)`
- `const char *(* json_object_iter_key_d)(void *iter)`
- `void *(* json_object_iter_next_d)(json_t *object, void *iter)`
- `int(* json_object_iter_set_d)(json_t *object, void *iter, json_t *value)`
- `int(* json_object_iter_set_new_d)(json_t *object, void *iter, json_t *value)`
- `json_t *(* json_object_iter_value_d)(void *iter)`
- `int(* json_object_set_d)(json_t *object, const char *key, json_t *value)`
- `int(* json_object_set_new_d)(json_t *object, const char *key, json_t *value)`
- `int(* json_object_set_new_nocheck_d)(json_t *object, const char *key, json_t *value)`
- `int(* json_object_set_nocheck_d)(json_t *object, const char *key, json_t *value)`
- `size_t(* json_object_size_d)(const json_t *object)`
- `int(* json_object_update_d)(json_t *object, json_t *other)`
- `json_t *(* json_pack_d)(const char *fmt,...)`
- `json_t *(* json_pack_ex_d)(json_error_t *error, size_t flags, const char *fmt,...)`
- `json_t *(* json_real_d)(double value)`
- `int(* json_real_set_d)(json_t *real, double value)`

- `double(* json_real_value_d)(const json_t *real)`
- `void(* json_set_alloc_funcs_d)(json_malloc_t malloc_fn, json_free_t free_fn)`
- `json_t *(* json_string_d)(const char *value)`
- `json_t *(* json_string_nocheck_d)(const char *value)`
- `int(* json_string_set_d)(json_t *string, const char *value)`
- `int(* json_string_set_nocheck_d)(json_t *string, const char *value)`
- `const char *(* json_string_value_d)(const json_t *string)`
- `json_t *(* json_true_d)(void)`
- `int(* json_unpack_d)(json_t *root, const char *fmt,...)`
- `int(* json_unpack_ex_d)(json_t *root, json_error_t *error, size_t flags, const char *fmt,...)`
- `json_t *(* json_vpack_ex_d)(json_error_t *error, size_t flags, const char *fmt, va_list ap)`
- `int(* json_vunpack_ex_d)(json_t *root, json_error_t *error, size_t flags, const char *fmt, va_list ap)`
- `const char *(* utf8proc_version_d)(void)`

UTF8.

- `const char *(* utf8proc_errmsg_d)(utf8proc_ssize_t errcode)`
- `const char *(* utf8proc_category_string_d)(utf8proc_int32_t c)`
- `utf8proc_category_t(* utf8proc_category_d)(utf8proc_int32_t c)`
- `const utf8proc_property_t *(* utf8proc_get_property_d)(utf8proc_int32_t uc)`
- `utf8proc_ssize_t(* utf8proc_iterate_d)(const utf8proc_uint8_t *str, utf8proc_ssize_t strlen, utf8proc_int32_t *codepoint_ref)`
- `char ** xmlParserVersion_d`

XML.

- `void(* xmlInitParser_d)(void)`
- `void(* xmlMemoryDump_d)(void)`
- `void(* xmlCleanupParser_d)(void)`
- `void(* xmlCleanupGlobals_d)(void)`
- `void(* xmlFreeDoc_d)(xmlDocPtr doc)`
- `void(* xmlFreeNode_d)(xmlNodePtr cur)`
- `xmlBufferPtr(* xmlBufferCreate_d)(void)`
- `void(* xmlBufferFree_d)(xmlBufferPtr buf)`
- `xmlParserCtxtPtr(* xmlNewParserCtxt_d)(void)`
- `int(* xmlBufferLength_d)(const xmlBufferPtr buf)`
- `void(* xmlFreeParserCtxt_d)(xmlParserCtxtPtr ctx)`
- `void(* xmlXPathFreeObject_d)(xmlXPathObjectPtr obj)`
- `void(* xmlXPathFreeContext_d)(xmlXPathContextPtr ctx)`
- `xmlXPathContextPtr(* xmlXPathNewContext_d)(xmlDocPtr doc)`
- `xmlNodePtr(* xmlNewNode_d)(xmlNsPtr ns, const xmlChar *name)`
- `const xmlChar *(* xmlBufferContent_d)(const xmlBufferPtr buf)`
- `xmlNodePtr(* xmlAddSibling_d)(xmlNodePtr cur, xmlNodePtr elem)`
- `int(* xmlNodeBufGetContent_d)(xmlBufferPtr buffer, xmlNodePtr cur)`
- `void(* xmlNodeSetContent_d)(xmlNodePtr cur, const xmlChar *content)`
- `xmlChar *(* xmlEncodeEntitiesReentrant_d)(xmlDocPtr doc, const xmlChar *input)`
- `void(* xmlDocDumpFormatMemory_d)(xmlDocPtr cur, xmlChar **mem, int *size, int format)`
- `xmlAttrPtr(* xmlSetProp_d)(xmlNodePtr node, const xmlChar *name, const xmlChar *value)`
- `xmlXPathObjectPtr(* xmlXPathEvalExpression_d)(const xmlChar *xpath, xmlXPathContextPtr ctx)`
- `int(* xmlXPathRegisterNs_d)(xmlXPathContextPtr ctx, const xmlChar *prefix, const xmlChar *ns_uri)`
- `xmlDocPtr(* xmlCtxtReadMemory_d)(xmlParserCtxtPtr ctx, const char *buffer, int size, const char *url, const char *encoding, int options)`
- `const char *(* zlibVersion_d)(void)`

ZLIB.

- `uLong(* compressBound_d)(uLong sourceLen)`
- `int(* uncompress_d)(Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen)`
- `int(* compress2_d)(Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen, int level)`

5.413.1 Define Documentation

5.413.1.1 `#define CONFIG_DEFAULT ""`

Definition at line 78 of file symbols.h.

5.413.1.2 `#define lint`

Definition at line 73 of file symbols.h.

5.413.1.3 `#define LOGDIR "~/"`

Definition at line 79 of file symbols.h.

5.413.1.4 `#define M_BIND(x)`

Value:

```
{ \
    .name = #x, \
    .pointer = (void *)&x##_d \
}
```

Definition at line 103 of file symbols.h.

Referenced by lib_load_bzip(), lib_load_cache(), lib_load_clamav(), lib_load_dkim(), lib_load_dspam(), lib_load_freetype(), lib_load_gd(), lib_load_jansson(), lib_load_jpeg(), lib_load_lzo(), lib_load_mysql(), lib_load_openssl(), lib_load_png(), lib_load_spf(), lib_load_tokyo(), lib_load_utf8proc(), lib_load_xml(), and lib_load_zlib().

5.413.2 Function Documentation

5.413.2.1 `STACK_OF(SSL_COMP)`

Definition at line 339 of file symbols.c.

References lib_magma, log_critical, NULLER, st_cmp_cs_eq(), tcversion_d, and xmlParserVersion_d.

Referenced by _do_ocsp_validation(), and _dump_ocsp_response_cb().

5.413.3 Variable Documentation

5.413.3.1 `int(* __lzo_init_v2_d)(unsigned, int, int, int, int, int, int, int, int)`

Referenced by lib_load_lzo().

5.413.3.2 `int(* ASN1_GENERALIZEDTIME_print_d)(BIO *fp, const ASN1_GENERALIZEDTIME *a)`

Referenced by _do_ocsp_validation(), and _dump_ocsp_response_cb().

5.413.3.3 `BIGNUM*(* ASN1_INTEGER_to_BN_d)(const ASN1_INTEGER *ai, BIGNUM *bn)`

Referenced by _dump_ocsp_response_cb().

5.413.3.4 unsigned char*(* ASN1_STRING_data_d)(ASN1_STRING *x)

Referenced by `_get_cert_subject_cn()`.

5.413.3.5 void(* ASN1_STRING_TABLE_cleanup_d)(void)

Referenced by `ssl_stop()`.

5.413.3.6 void(* BIO_free_all_d)(BIO *a)

Referenced by `_do_ocsp_validation()`.

5.413.3.7 int(* BIO_free_d)(BIO *a)

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.413.3.8 BIO*(* BIO_new_fp_d)(FILE *stream, int close_flag)

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.413.3.9 BIO*(* BIO_new_socket_d)(int sock, int close_flag)

Referenced by `_do_ocsp_validation()`, `tls_client_alloc()`, and `tls_server_alloc()`.

5.413.3.10 void(* BIO_sock_cleanup_d)(void)

Referenced by `ssl_stop()`.

5.413.3.11 int(* BIO_vprintf_d)(BIO *bio, const char *format, va_list args)**5.413.3.12 BIGNUM*(* BN_bin2bn_d)(const unsigned char *s, int len, BIGNUM *ret)**

Referenced by `_decode_rsa_pubkey()`, `_get_rsa_dnskey()`, `deprecated_ecies_key_private()`, `dh_params_2048()`, `dh_params_4096()`, `ecies_key_private()`, and `secp256k1_private_set()`.

5.413.3.13 int(* BN_bn2bin_d)(const BIGNUM *, unsigned char *)

Referenced by `_encode_rsa_pubkey()`, `deprecated_ecies_key_private_bin()`, `ecies_key_private_bin()`, and `secp256k1_private_get()`.

5.413.3.14 char*(* BN_bn2dec_d)(const BIGNUM *a)**5.413.3.15 char*(* BN_bn2hex_d)(const BIGNUM *a)**

Referenced by `_dump_ocsp_response_cb()`, `deprecated_ecies_key_private_hex()`, and `ecies_key_private_hex()`.

5.413.3.16 `int(* BN_bn2mpi_d)(const BIGNUM *a, unsigned char *to)`

5.413.3.17 `int(* BN_cmp_d)(const BIGNUM *a, const BIGNUM *b)`

5.413.3.18 `void(* BN_CTX_free_d)(BN_CTX *ctx)`

Referenced by `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.413.3.19 `BN_CTX*(* BN_CTX_new_d)(void)`

Referenced by `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.413.3.20 `void(* BN_CTX_start_d)(BN_CTX *ctx)`

Referenced by `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.413.3.21 `void(* BN_free_d)(BIGNUM *a)`

Referenced by `_decode_rsa_pubkey()`, `_dump_ocsp_response_cb()`, `_get_rsa_dnskey()`, `deprecated_ecies_key_private()`, `ecies_key_private()`, and `secp256k1_private_set()`.

5.413.3.22 `int(* BN_hex2bn_d)(BIGNUM **a, const char *str)`

Referenced by `deprecated_ecies_key_private()`, and `ecies_key_private()`.

5.413.3.23 `BIGNUM*(* BN_mpi2bn_d)(unsigned char *s, int len, BIGNUM *ret)`

5.413.3.24 `int(* BN_num_bits_d)(const BIGNUM *)`

Referenced by `_encode_rsa_pubkey()`, `_get_rsa_dnskey()`, and `secp256k1_private_get()`.

5.413.3.25 `size_t(* BUF_strlcat_d)(char *dst, const char *src, size_t siz)`

Referenced by `_push_error_stack_openssl()`.

5.413.3.26 `int(* BZ2_bzBuffToBuffCompress_d)(char *dest, unsigned int *destLen, char *source, unsigned int sourceLen, int blockSize100k, int verbosity, int workFactor)`

Referenced by `compress_bzip()`.

5.413.3.27 `int(* BZ2_bzBuffToBuffDecompress_d)(char *dest, unsigned int *destLen, char *source, unsigned int sourceLen, int small, int verbosity)`

Referenced by `decompress_bzip()`.

5.413.3.28 `const char*(* BZ2_bzlibVersion_d)(void)`

BZIP.

Referenced by `lib_load_bzip()`.

5.413.3.29 `int(* cl_countsigs_d)(const char *path, unsigned int countoptions, unsigned int *sigs)`

Referenced by `virus_sigs_total()`.

5.413.3.30 `int(* cl_engine_compile_d)(struct cl_engine *engine)`

Referenced by `virus_engine_create()`.

5.413.3.31 `int(* cl_engine_free_d)(struct cl_engine *engine)`

Referenced by `virus_engine_create()`, and `virus_engine_destroy()`.

5.413.3.32 `struct cl_engine*(* cl_engine_new_d)(void)`

Referenced by `virus_engine_create()`.

5.413.3.33 `int(* cl_engine_set_num_d)(struct cl_engine *engine, enum cl_engine_field field, long long num)`

Referenced by `virus_engine_create()`.

5.413.3.34 `int(* cl_engine_set_str_d)(struct cl_engine *engine, enum cl_engine_field field, const char *str)`

Referenced by `virus_engine_create()`.

5.413.3.35 `int(* cl_init_d)(unsigned int initoptions)`

Referenced by `virus_start()`.

5.413.3.36 `int(* cl_load_d)(const char *path, struct cl_engine *engine, unsigned int *signo, unsigned int dboptions)`

Referenced by `virus_engine_create()`.

5.413.3.37 `const char*(* cl_retver_d)(void)`

Referenced by `lib_version_clamav()`.

5.413.3.38 `int(* cl_scandesc_d)(int desc, const char **virname, unsigned long int *scanned, const struct cl_engine *engine, unsigned int scanoptions)`

Referenced by `virus_check()`.

5.413.3.39 `void(* cl_shutdown_d)(void)`

CLAMAV.

Referenced by `virus_stop()`.

5.413.3.40 `int(* cl_statchkdir_d)(const struct cl_stat *dbstat)`

Referenced by `virus_engine_refresh()`.

5.413.3.41 int(* cl_statfree_d)(struct cl_stat *dbstat)

Referenced by virus_engine_refresh(), virus_start(), and virus_stop().

5.413.3.42 int(* cl_statinidir_d)(const char *dirname, struct cl_stat *dbstat)

Referenced by virus_engine_refresh(), and virus_start().

5.413.3.43 const char*(* cl_strerror_d)(int clerror)

Referenced by virus_check(), virus_engine_create(), virus_sigs_total(), and virus_start().

5.413.3.44 void(* COMP_zlib_cleanup_d)(void)

Referenced by ssl_stop().

5.413.3.45 int(* compress2_d)(Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen, int level)

Referenced by compress_zlib().

5.413.3.46 uLong(* compressBound_d)(uLong sourceLen)

Referenced by compress_zlib().

5.413.3.47 void(* CONF_modules_unload_d)(int all)

Referenced by ssl_stop().

5.413.3.48 void(* CRYPTO_cleanup_all_ex_data_d)(void)

Referenced by ssl_stop().

5.413.3.49 void(* CRYPTO_free_d)(void *)

Referenced by _get_cache_ocsp_id(), and _get_x509_cert_sha_hash().

5.413.3.50 int(* CRYPTO_num_locks_d)(void)

Referenced by ssl_start(), and ssl_stop().

5.413.3.51 void(* CRYPTO_set_id_callback_d)(unsigned long(*id_function)(void))

Referenced by ssl_start(), and ssl_stop().

5.413.3.52 int(* CRYPTO_set_locked_mem_functions_d)(void *(*m)(size_t), void(*free_func)(void *))

Referenced by ssl_start().

5.413.3.53 `void(* CRYPTO_set_locking_callback_d)(void(*locking_function)(int mode, int n, const char *file, int line))`

Referenced by `ssl_start()`, and `ssl_stop()`.

5.413.3.54 `int(* CRYPTO_set_mem_functions_d)(void *(*m)(size_t), void *(*r)(void *, size_t), void(*f)(void *))`

5.413.3.55 `ECDSA_SIG>(* d2i_ECDSA_SIG_d)(ECDSA_SIG **sig, const unsigned char **pp, long len)`

Referenced by `_verify_ec_signature()`.

5.413.3.56 `EC_KEY>(* d2i_ECPrivateKey_d)(EC_KEY **key, const unsigned char **in, long len)`

Referenced by `encrypt_deserialize_privkey()`.

5.413.3.57 `OCSP_RESPONSE(* d2i_OCSP_RESPONSE_d)(OCSP_RESPONSE **a, const unsigned char **in, long len)`

Referenced by `_deserialize_ocsp_response_cb()`, and `_ocsp_response_callback()`.

5.413.3.58 `int(* DH_check_d)(const DH *dh, int *ret)`

Referenced by `dh_exchange_2048()`, `dh_exchange_4096()`, and `dh_params_generate()`.

5.413.3.59 `void(* DH_free_d)(DH *dh)`

Referenced by `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_params_2048()`, `dh_params_4096()`, and `dh_params_generate()`.

5.413.3.60 `int(* DH_generate_parameters_ex_d)(DH *dh, int prime_len, int generator, BN_GENCB *cb)`

Referenced by `dh_exchange_2048()`, `dh_exchange_4096()`, and `dh_params_generate()`.

5.413.3.61 `DH(* DH_new_d)(void)`

OPENSSL.

Referenced by `dh_exchange_2048()`, `dh_exchange_4096()`, `dh_params_2048()`, `dh_params_4096()`, and `dh_params_generate()`.

5.413.3.62 `DKIM_STAT(* dkim_body_d)(DKIM *dkim, u_char *buf, size_t len)`

5.413.3.63 `DKIM_STAT(* dkim_chunk_d)(DKIM *dkim, unsigned char *chunkp, size_t len)`

Referenced by `dkim_signature_create()`, and `dkim_signature_verify()`.

5.413.3.64 `void(* dkim_close_d)(DKIM_LIB *lib)`

Referenced by `dkim_stop()`.

5.413.3.65 `DKIM_STAT(* dkim_eoh_d)(DKIM *dkim)`

DKIM Note that `dkim_getsighdr_d` is used by the library, so were using `dkim_getsighdrx_d`. Note that `dkim_test_dns_put_d` is only used by the verification unit test, to load public keys which may no longer be available on the public internet.

DKIM

Note:

that dkim_getsighdr_d is used by the library, so were using dkim_getsighdrx_d.

5.413.3.66 DKIM_STAT(* dkim_eom_d)(DKIM *dkim, _Bool *testkey)

Referenced by dkim_signature_create(), and dkim_signature_verify().

5.413.3.67 DKIM_STAT(* dkim_free_d)(DKIM *dkim)

Referenced by dkim_signature_create(), and dkim_signature_verify().

5.413.3.68 char*(* dkim_geterror_d)(DKIM *dkim)

Referenced by dkim_signature_create().

5.413.3.69 const char*(* dkim_getresultstr_d)(DKIM_STAT result)

Referenced by dkim_signature_create(), and dkim_signature_verify().

5.413.3.70 DKIM_STAT(* dkim_getsighdrx_d)(DKIM *dkim, u_char *buf, size_t len, size_t initial)

Referenced by dkim_signature_create(), and lib_load_dkim().

5.413.3.71 DKIM_STAT(* dkim_header_d)(DKIM *dkim, u_char *hdr, size_t len)

5.413.3.72 DKIM_LIB(* dkim_init_d)(void *(*mallocf)(void *closure, size_t nbytes), void(*freef)(void *closure, void *p))

Referenced by dkim_start().

5.413.3.73 uint32_t(* dkim_libversion_d)(void)

Referenced by lib_load_dkim().

5.413.3.74 void(* dkim_mfree_d)(DKIM_LIB *libhandle, void *closure, void *ptr)

5.413.3.75 DKIM*(* dkim_sign_d)(DKIM_LIB *libhandle, const unsigned char *id, void *memclosure, const dkim_sigkey_t secretkey, const unsigned char *selector, const unsigned char *domain, dkim_canon_t hdr_canon_alg, dkim_canon_t body_canon_alg, dkim_alg_t sign_alg, off_t length, DKIM_STAT *statp)

Referenced by dkim_signature_create().

5.413.3.76 int(* dkim_test_dns_put_d)(DKIM *dkim, int class, int type, int prec, u_char *name, u_char *data)

5.413.3.77 DKIM*(* dkim_verify_d)(DKIM_LIB *libhandle, const unsigned char *id, void *memclosure, DKIM_STAT *statp)

Referenced by dkim_signature_verify().

5.413.3.78 `int(* dspam_attach_d)(DSPAM_CTX *CTX, void *dbh)`

Referenced by `dspam_check()`, and `dspam_train()`.

5.413.3.79 `DSPAM_CTX*(* dspam_create_d)(const char *username, const char *group, const char *home, int operating_mode, u_int32_t flags)`

Referenced by `dspam_check()`, and `dspam_train()`.

5.413.3.80 `void(* dspam_destroy_d)(DSPAM_CTX *CTX)`

Referenced by `dspam_check()`, and `dspam_train()`.

5.413.3.81 `int(* dspam_detach_d)(DSPAM_CTX *CTX)`

Referenced by `dspam_check()`, and `dspam_train()`.

5.413.3.82 `int(* dspam_init_driver_d)(DRIVER_CTX *DTX)`

Referenced by `dspam_start()`.

5.413.3.83 `int(* dspam_process_d)(DSPAM_CTX *CTX, const char *message)`

Referenced by `dspam_check()`, and `dspam_train()`.

5.413.3.84 `int(* dspam_shutdown_driver_d)(DRIVER_CTX *DTX)`

Referenced by `dspam_stop()`.

5.413.3.85 `const char*(* dspam_version_d)(void)`

DSPAM.

Referenced by `lib_version_dspam()`.

5.413.3.86 `void(* EC_GROUP_clear_free_d)(EC_GROUP *group)`

Referenced by `_crypto_shutdown()`, `encrypt_ctx_free()`, and `encrypt_ctx_new()`.

5.413.3.87 `void(* EC_GROUP_free_d)(EC_GROUP *group)`

Referenced by `deprecated_ecies_group()`, `deprecated_ecies_stop()`, `ecies_group()`, `ecies_stop()`, `prime_start()`, and `prime_stop()`.

5.413.3.88 `EC_GROUP*(* EC_GROUP_new_by_curve_name_d)(int nid)`

Referenced by `_crypto_init()`, `_deserialize_ec_pubkey()`, `deprecated_ecies_group()`, `ecies_group()`, `encrypt_ctx_new()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, and `prime_start()`.

5.413.3.89 int(* EC_GROUP_precompute_mult_d)(EC_GROUP *group, BN_CTX *ctx)

Referenced by deprecated_ecies_group(), ecies_group(), and prime_start().

5.413.3.90 void(* EC_GROUP_set_point_conversion_form_d)(EC_GROUP *, point_conversion_form_t)

Referenced by deprecated_ecies_group(), ecies_group(), prime_start(), and ssl_ecdh_exchange_callback().

5.413.3.91 int(* EC_KEY_check_key_d)(const EC_KEY *key)

Referenced by deprecated_ecies_key_public(), ecies_key_public(), and secp256k1_private_set().

5.413.3.92 void(* EC_KEY_free_d)(EC_KEY *key)

Referenced by _deserialize_ec_pubkey(), _free_ec_key(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_ecies_key_alloc(), deprecated_ecies_key_create(), deprecated_ecies_key_free(), deprecated_ecies_key_private(), deprecated_ecies_key_public(), ecies_decrypt(), ecies_encrypt(), ecies_key_alloc(), ecies_key_create(), ecies_key_free(), ecies_key_private(), ecies_key_public(), encrypt_deserialize_privkey(), encrypt_deserialize_pubkey(), encrypt_keypair_generate(), secp256k1_alloc(), secp256k1_free(), secp256k1_generate(), secp256k1_private_set(), secp256k1_public_set(), and ssl_stop().

5.413.3.93 int(* EC_KEY_generate_key_d)(EC_KEY *key)

Referenced by deprecated_ecies_key_create(), ecies_key_create(), encrypt_keypair_generate(), and secp256k1_generate().

5.413.3.94 const EC_GROUP*(* EC_KEY_get0_group_d)(const EC_KEY *key)

Referenced by deprecated_ecies_encrypt(), deprecated_ecies_key_public(), deprecated_ecies_key_public_bin(), deprecated_ecies_key_public_hex(), ecies_encrypt(), ecies_key_public(), ecies_key_public_bin(), ecies_key_public_hex(), secp256k1_private_set(), secp256k1_public_get(), secp256k1_public_set(), and ssl_ecdh_exchange_callback().

5.413.3.95 const BIGNUM*(* EC_KEY_get0_private_key_d)(const EC_KEY *key)

Referenced by deprecated_ecies_key_private_bin(), deprecated_ecies_key_private_hex(), ecies_key_private_bin(), ecies_key_private_hex(), secp256k1_private_get(), and secp256k1_type().

5.413.3.96 const EC_POINT*(* EC_KEY_get0_public_key_d)(const EC_KEY *key)

Referenced by _compute_aes256_kek(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_ecies_key_public_bin(), deprecated_ecies_key_public_hex(), ecies_decrypt(), ecies_encrypt(), ecies_key_public_bin(), ecies_key_public_hex(), secp256k1_compute_kek(), secp256k1_public_get(), and secp256k1_type().

5.413.3.97 point_conversion_form_t(* EC_KEY_get_conv_form_d)(const EC_KEY *key)

Referenced by secp256k1_public_get().

5.413.3.98 EC_KEY*(* EC_KEY_new_by_curve_name_d)(int nid)

Referenced by deprecated_ecies_key_alloc(), ecies_key_alloc(), secp256k1_alloc(), and ssl_ecdh_exchange_callback().

5.413.3.99 EC_KEY*(* EC_KEY_new_d)(void)

Referenced by `_deserialize_ec_pubkey()`, `deprecated_ecies_key_alloc()`, `ecies_key_alloc()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, `encrypt_keypair_generate()`, and `secp256k1_alloc()`.

5.413.3.100 void(* EC_KEY_set_conv_form_d)(EC_KEY *eckey, point_conversion_form_t cform)

Referenced by `deprecated_ecies_key_alloc()`, `ecies_key_alloc()`, and `secp256k1_alloc()`.

5.413.3.101 int(* EC_KEY_set_group_d)(EC_KEY *key, const EC_GROUP *group)

Referenced by `_deserialize_ec_pubkey()`, `deprecated_ecies_key_alloc()`, `ecies_key_alloc()`, `encrypt_deserialize_privkey()`, `encrypt_deserialize_pubkey()`, `encrypt_keypair_generate()`, and `secp256k1_alloc()`.

5.413.3.102 int(* EC_KEY_set_private_key_d)(EC_KEY *key, const BIGNUM *prv)

Referenced by `deprecated_ecies_key_private()`, `ecies_key_private()`, and `secp256k1_private_set()`.

5.413.3.103 int(* EC_KEY_set_public_key_d)(EC_KEY *key, const EC_POINT *pub)

Referenced by `deprecated_ecies_key_public()`, `ecies_key_public()`, `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.413.3.104 int(* EC_POINT_cmp_d)(const EC_GROUP *group, const EC_POINT *a, const EC_POINT *b, BN_CTX *ctx)**5.413.3.105 void(* EC_POINT_free_d)(EC_POINT *point)**

Referenced by `deprecated_ecies_key_public()`, `ecies_key_public()`, `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.413.3.106 EC_POINT*(* EC_POINT_hex2point_d)(const EC_GROUP *, const char *, EC_POINT *, BN_CTX *)

Referenced by `deprecated_ecies_key_public()`, and `ecies_key_public()`.

5.413.3.107 int(* EC_POINT_mul_d)(const EC_GROUP *group, EC_POINT *r, const BIGNUM *g_scalar, const EC_POINT *point, const BIGNUM *p_scalar, BN_CTX *ctx)

Referenced by `secp256k1_private_set()`.

5.413.3.108 EC_POINT*(* EC_POINT_new_d)(const EC_GROUP *group)

Referenced by `deprecated_ecies_key_public()`, `ecies_key_public()`, `secp256k1_private_set()`, and `secp256k1_public_set()`.

5.413.3.109 int(* EC_POINT_oct2point_d)(const EC_GROUP *group, EC_POINT *p, const unsigned char *buf, size_t len, BN_CTX *ctx)

Referenced by `deprecated_ecies_key_public()`, `ecies_key_public()`, and `secp256k1_public_set()`.

5.413.3.110 char*(* EC_POINT_point2hex_d)(const EC_GROUP *, const EC_POINT *, point_conversion_form_t form, BN_CTX *)

Referenced by `deprecated_ecies_key_public_hex()`, and `ecies_key_public_hex()`.

5.413.3.111 `size_t(* EC_POINT_point2oct_d)(const EC_GROUP *group, const EC_POINT *p, point_conversion_form_t form, unsigned char *buf, size_t len, BN_CTX *ctx)`

Referenced by `deprecated_ecies_encrypt()`, `deprecated_ecies_key_public_bin()`, `ecies_encrypt()`, `ecies_key_public_bin()`, and `secp256k1_public_get()`.

5.413.3.112 `int(* ECDH_compute_key_d)(void *out, size_t outlen, const EC_POINT *pub_key, EC_KEY *ecdh, void *(*KDF)(const void *in, size_t inlen, void *out, size_t *outlen))`

Referenced by `_compute_aes256_kek()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `ecies_decrypt()`, `ecies_encrypt()`, and `secp256k1_compute_kek()`.

5.413.3.113 `ECDSA_SIG>(* ECDSA_do_sign_d)(const unsigned char *dgst, int dgst_len, EC_KEY *eckey)`

Referenced by `_ec_sign_data()`.

5.413.3.114 `int(* ECDSA_do_verify_d)(const unsigned char *dgst, int dgst_len, const ECDSA_SIG *sig, EC_KEY *eckey)`

Referenced by `_verify_ec_signature()`.

5.413.3.115 `void(* ECDSA_SIG_free_d)(ECDSA_SIG *a)`

Referenced by `_ec_sign_data()`, and `_verify_ec_signature()`.

5.413.3.116 `void(* ED25519_keypair_d)(uint8_t out_public_key[32], uint8_t out_private_key[64])`

Referenced by `ed25519_generate()`.

5.413.3.117 `void(* ED25519_keypair_from_seed_d)(uint8_t out_public_key[32], uint8_t out_private_key[64], const uint8_t seed[32])`

Referenced by `ed25519_private_set()`.

5.413.3.118 `int(* ED25519_sign_d)(uint8_t *out_sig, const uint8_t *message, size_t message_len, const uint8_t private_key[64])`

Referenced by `ed25519_sign()`.

5.413.3.119 `int(* ED25519_verify_d)(const uint8_t *message, size_t message_len, const uint8_t signature[64], const uint8_t public_key[32])`

Referenced by `ed25519_verify()`.

5.413.3.120 `void(* ENGINE_cleanup_d)(void)`

Referenced by `ssl_stop()`.

5.413.3.121 `void(* ERR_clear_error_d)(void)`

Referenced by `tls_client_alloc()`, `tls_read()`, `tls_server_alloc()`, and `tls_write()`.

5.413.3.122 `void(* ERR_error_string_n_d)(unsigned long e, char *buf, size_t len)`

Referenced by `_push_error_stack_openssl()`, `ssl_error_string()`, `tls_continue()`, and `tls_error()`.

5.413.3.123 `void(* ERR_free_strings_d)(void)`

Referenced by `_crypto_shutdown()`, `_ssl_shutdown()`, `encrypt_ctx_free()`, `encrypt_ctx_new()`, and `ssl_stop()`.

5.413.3.124 `unsigned long(* ERR_get_error_d)(void)`

Referenced by `_push_error_stack_openssl()`, `ssl_error_string()`, `tls_continue()`, and `tls_error()`.

5.413.3.125 `void(* ERR_load_crypto_strings_d)(void)`

Referenced by `_push_error_stack_openssl()`.

5.413.3.126 `unsigned long(* ERR_peek_error_d)(void)`

5.413.3.127 `unsigned long(* ERR_peek_error_line_data_d)(const char **file, int *line, const char **data, int *flags)`

Referenced by `_push_error_stack_openssl()`.

5.413.3.128 `void(* ERR_print_errors_fp_d)(FILE *fp)`

Referenced by `_do_ocsp_validation()`, `_dump_ocsp_response_cb()`, `_ocsp_response_callback()`, `_ssl_disconnect()`, `_ssl_fd_loop()`, and `_verify_certificate_callback()`.

5.413.3.129 `void(* ERR_put_error_d)(int lib, int func, int reason, const char *file, int line)`

5.413.3.130 `void(* ERR_remove_thread_state_d)(const CRYPTO_THREADID *tid)`

Referenced by `ssl_stop()`, `ssl_thread_stop()`, and `tls_free()`.

5.413.3.131 `const EVP_CIPHER*(* EVP_aes_256_cbc_d)(void)`

Referenced by `_decrypt_aes_256()`, and `_encrypt_aes_256()`.

5.413.3.132 `const EVP_CIPHER*(* EVP_aes_256_gcm_d)(void)`

Referenced by `aes_artifact_decrypt()`, `aes_artifact_encrypt()`, `aes_chunk_decrypt()`, `aes_chunk_encrypt()`, `stacie_decrypt()`, and `stacie_encrypt()`.

5.413.3.133 `int(* EVP_CIPHER_block_size_d)(const EVP_CIPHER *cipher)`

Referenced by `cipher_block_length()`, `deprecated_ecies_encrypt()`, and `ecies_encrypt()`.

5.413.3.134 `int(* EVP_CIPHER_CTX_block_size_d)(const EVP_CIPHER_CTX *ctx)`

Referenced by `deprecated_symmetric_decrypt()`, `deprecated_symmetric_encrypt()`, `symmetric_decrypt()`, and `symmetric_encrypt()`.

5.413.3.135 int(* EVP_CIPHER_CTX_cleanup_d)(EVP_CIPHER_CTX *a)

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), ecies_decrypt(), ecies_encrypt(), stacie_decrypt(), stacie_encrypt(), symmetric_decrypt(), and symmetric_encrypt().

5.413.3.136 int(* EVP_CIPHER_CTX_ctrl_d)(EVP_CIPHER_CTX *ctx, int type, int arg, void *ptr)

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), stacie_decrypt(), stacie_encrypt(), symmetric_decrypt(), and symmetric_encrypt().

5.413.3.137 unsigned long(* EVP_CIPHER_CTX_flags_d)(const EVP_CIPHER_CTX *ctx)**5.413.3.138 void(* EVP_CIPHER_CTX_free_d)(EVP_CIPHER_CTX *a)**

Referenced by _decrypt_aes_256(), and _encrypt_aes_256().

5.413.3.139 void(* EVP_CIPHER_CTX_init_d)(EVP_CIPHER_CTX *a)

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), ecies_decrypt(), ecies_encrypt(), stacie_decrypt(), stacie_encrypt(), symmetric_decrypt(), and symmetric_encrypt().

5.413.3.140 int(* EVP_CIPHER_CTX_iv_length_d)(const EVP_CIPHER_CTX *ctx)

Referenced by deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), symmetric_decrypt(), and symmetric_encrypt().

5.413.3.141 int(* EVP_CIPHER_CTX_key_length_d)(const EVP_CIPHER_CTX *ctx)

Referenced by aes_artifact_decrypt(), aes_artifact_encrypt(), aes_chunk_decrypt(), aes_chunk_encrypt(), deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), stacie_decrypt(), stacie_encrypt(), symmetric_decrypt(), and symmetric_encrypt().

5.413.3.142 EVP_CIPHER_CTX*(* EVP_CIPHER_CTX_new_d)(void)

Referenced by _decrypt_aes_256(), and _encrypt_aes_256().

5.413.3.143 int(* EVP_CIPHER_CTX_set_padding_d)(EVP_CIPHER_CTX *c, int pad)

Referenced by _decrypt_aes_256(), _encrypt_aes_256(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), ecies_decrypt(), and ecies_encrypt().

5.413.3.144 unsigned long(* EVP_CIPHER_flags_d)(const EVP_CIPHER *cipher)

Referenced by deprecated_symmetric_decrypt(), deprecated_symmetric_encrypt(), symmetric_decrypt(), and symmetric_encrypt().

5.413.3.145 int(* EVP_CIPHER_iv_length_d)(const EVP_CIPHER *cipher)

Referenced by cipher_vector_length().

5.413.3.146 int(* EVP_CIPHER_key_length_d)(const EVP_CIPHER *cipher)

Referenced by cipher_key_length(), deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), ecies_decrypt(), and ecies_encrypt().

5.413.3.147 int(* EVP_CIPHER_nid_d)(const EVP_CIPHER *cipher)

Referenced by cipher_block_length(), cipher_key_length(), cipher_numeric_id(), and cipher_vector_length().

5.413.3.148 void(* EVP_cleanup_d)(void)

Referenced by _crypto_shutdown(), encrypt_ctx_free(), encrypt_ctx_new(), and ssl_stop().

5.413.3.149 int(* EVP_DecryptFinal_ex_d)(EVP_CIPHER_CTX *ctx, unsigned char *outm, int *outl)

Referenced by _decrypt_aes_256(), aes_artifact_decrypt(), aes_chunk_decrypt(), deprecated_ecies_decrypt(), deprecated_symmetric_decrypt(), ecies_decrypt(), stacie_decrypt(), and symmetric_decrypt().

5.413.3.150 int(* EVP_DecryptInit_ex_d)(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *impl, const unsigned char *key, const unsigned char *iv)

Referenced by _decrypt_aes_256(), aes_artifact_decrypt(), aes_chunk_decrypt(), deprecated_ecies_decrypt(), deprecated_symmetric_decrypt(), ecies_decrypt(), stacie_decrypt(), and symmetric_decrypt().

5.413.3.151 int(* EVP_DecryptUpdate_d)(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl, const unsigned char *in, int inl)

Referenced by _decrypt_aes_256(), aes_artifact_decrypt(), aes_chunk_decrypt(), deprecated_ecies_decrypt(), deprecated_symmetric_decrypt(), ecies_decrypt(), stacie_decrypt(), and symmetric_decrypt().

5.413.3.152 int(* EVP_Digest_d)(const void *data, size_t count, unsigned char *md, unsigned int *size, const EVP_MD *type, ENGINE *impl)

Referenced by _ecies_env_derivation(), deprecated_ecies_envelope_derivation(), and ecies_envelope_derivation().

5.413.3.153 int(* EVP_DigestFinal_d)(EVP_MD_CTX *ctx, unsigned char *md, unsigned int *s)

Referenced by hash_digest(), stacie_derive_key(), stacie_derive_token(), and stacie_realm_key().

5.413.3.154 int(* EVP_DigestFinal_ex_d)(EVP_MD_CTX *ctx, unsigned char *md, unsigned int *s)**5.413.3.155 int(* EVP_DigestInit_d)(EVP_MD_CTX *ctx, const EVP_MD *type)**

Referenced by _rsa_verify_record().

5.413.3.156 int(* EVP_DigestInit_ex_d)(EVP_MD_CTX *ctx, const EVP_MD *type, ENGINE *impl)

Referenced by hash_digest(), stacie_derive_key(), stacie_derive_token(), and stacie_realm_key().

5.413.3.157 int(* EVP_DigestUpdate_d)(EVP_MD_CTX *ctx, const void *d, size_t cnt)

Referenced by _rsa_verify_record(), hash_digest(), stacie_derive_key(), stacie_derive_token(), and stacie_realm_key().

5.413.3.158 int(* EVP_EncryptFinal_ex_d)(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)

Referenced by `_encrypt_aes_256()`, `aes_artifact_encrypt()`, `aes_chunk_encrypt()`, `deprecated_ecies_encrypt()`, `deprecated_symmetric_encrypt()`, `ecies_encrypt()`, `stacie_encrypt()`, and `symmetric_encrypt()`.

5.413.3.159 int(* EVP_EncryptInit_ex_d)(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *impl, const unsigned char *key, const unsigned char *iv)

Referenced by `_encrypt_aes_256()`, `aes_artifact_encrypt()`, `aes_chunk_encrypt()`, `deprecated_ecies_encrypt()`, `deprecated_symmetric_encrypt()`, `ecies_encrypt()`, `stacie_encrypt()`, and `symmetric_encrypt()`.

5.413.3.160 int(* EVP_EncryptUpdate_d)(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl, const unsigned char *in, int inl)

Referenced by `_encrypt_aes_256()`, `aes_artifact_encrypt()`, `aes_chunk_encrypt()`, `deprecated_ecies_encrypt()`, `deprecated_symmetric_encrypt()`, `ecies_encrypt()`, `stacie_encrypt()`, and `symmetric_encrypt()`.

5.413.3.161 const EVP_CIPHER*(* EVP_get_cipherbyname_d)(const char *name)

Referenced by `cipher_id()`, `cipher_name()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_start()`, `ecies_decrypt()`, `ecies_encrypt()`, and `ecies_start()`.

5.413.3.162 const EVP_MD*(* EVP_get_digestbyname_d)(const char *name)

Referenced by `_crypto_init()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_start()`, `digest_id()`, `digest_name()`, `ecies_decrypt()`, `ecies_encrypt()`, `ecies_start()`, and `encrypt_ctx_new()`.

5.413.3.163 const EVP_MD*(* EVP_md4_d)(void)

Referenced by `deprecated_hmac_md4()`, `hash_md4()`, and `hmac_md4()`.

5.413.3.164 const EVP_MD*(* EVP_md5_d)(void)

Referenced by `deprecated_hmac_md5()`, `hash_md5()`, and `hmac_md5()`.

5.413.3.165 int(* EVP_MD_CTX_cleanup_d)(EVP_MD_CTX *ctx)

Referenced by `hash_digest()`, `stacie_derive_key()`, `stacie_derive_token()`, and `stacie_realm_key()`.

5.413.3.166 void(* EVP_MD_CTX_init_d)(EVP_MD_CTX *ctx)

Referenced by `_rsa_verify_record()`, `hash_digest()`, `stacie_derive_key()`, `stacie_derive_token()`, and `stacie_realm_key()`.

5.413.3.167 int(* EVP_MD_size_d)(const EVP_MD *md)

Referenced by `deprecated_ecies_encrypt()`, `deprecated_hmac_digest()`, `digest_length_output()`, `ecies_encrypt()`, `hash_digest()`, and `hmac_digest()`.

5.413.3.168 int(* EVP_MD_type_d)(const EVP_MD *md)

Referenced by `digest_length_output()`.

5.413.3.169 EVP_PKEY*(* EVP_PKEY_new_d)(void)

Referenced by `_rsa_verify_record()`.

5.413.3.170 int(* EVP_PKEY_set1_RSA_d)(EVP_PKEY *pkey, struct rsa_st *key)

Referenced by `_rsa_verify_record()`.

5.413.3.171 const EVP_MD*(* EVP_ripemd160_d)(void)

Referenced by `deprecated_hmac_ripemd160()`, `hash_ripemd160()`, and `hmac_ripemd160()`.

5.413.3.172 const EVP_MD*(* EVP_sha1_d)(void)

Referenced by `_rsa_verify_record()`, `deprecated_hmac_sha1()`, `hash_sha1()`, and `hmac_sha1()`.

5.413.3.173 const EVP_MD*(* EVP_sha224_d)(void)

Referenced by `deprecated_hmac_sha224()`, `hash_sha224()`, and `hmac_sha224()`.

5.413.3.174 const EVP_MD*(* EVP_sha256_d)(void)

Referenced by `_rsa_verify_record()`, `deprecated_hmac_sha256()`, `hash_sha256()`, and `hmac_sha256()`.

5.413.3.175 const EVP_MD*(* EVP_sha384_d)(void)

Referenced by `deprecated_hmac_sha384()`, `hash_sha384()`, and `hmac_sha384()`.

5.413.3.176 const EVP_MD*(* EVP_sha512_d)(void)

Referenced by `_rsa_verify_record()`, `deprecated_hmac_sha512()`, `hash_sha512()`, `hmac_sha512()`, `stacie_derive_key()`, `stacie_derive_seed()`, `stacie_derive_token()`, and `stacie_realms_key()`.

5.413.3.177 const EVP_MD*(* EVP_sha_d)(void)

Referenced by `deprecated_hmac_sha()`, `hash_sha()`, and `hmac_sha()`.

5.413.3.178 int(* EVP_VerifyFinal_d)(EVP_MD_CTX *ctx, const unsigned char *sigbuf, unsigned int siglen, EVP_PKEY *pkey)

Referenced by `_rsa_verify_record()`.

5.413.3.179 FT_Error(* FT_Done_FreeType_d)(FT_Library library)

FreeType.

Referenced by `lib_version_freetype()`.

5.413.3.180 FT_Error(* FT_Init_FreeType_d)(FT_Library *alibrary)

Referenced by `lib_version_freetype()`.

5.413.3.181 void(* FT_Library_Version_d)(FT_Library library, FT_Int *amajor, FT_Int *aminor, FT_Int *apatch)

Referenced by lib_version_freetype().

5.413.3.182 void(* gdFree_d)(void *m)

Referenced by register_captcha_generate().

5.413.3.183 int(* gdImageColorResolve_d)(gdImagePtr im, int r, int g, int b)

Referenced by register_captcha_generate(), and register_captcha_write_noise().

5.413.3.184 gdImagePtr(* gdImageCreate_d)(int sx, int sy)

Referenced by register_captcha_generate().

5.413.3.185 void(* gdImageDestroy_d)(gdImagePtr im)

Referenced by register_captcha_generate().

5.413.3.186 void*(* gdImageGifPtr_d)(gdImagePtr im, int *size)

Referenced by register_captcha_generate().

5.413.3.187 void*(* gdImageJpegPtr_d)(gdImagePtr im, int *size, int quality)

5.413.3.188 void(* gdImageSetPixel_d)(gdImagePtr im, int x, int y, int color)

Referenced by register_captcha_write_noise().

5.413.3.189 char*(* gdImageStringFT_d)(gdImage *im, int *brext, int fg, char *fontlist, double psize, double angle, int x, int y, char *string)

Referenced by register_captcha_generate().

5.413.3.190 const char*(* gdVersionString_d)(void)

GD.

Referenced by lib_version_gd().

5.413.3.191 void(* HMAC_CTX_cleanup_d)(HMAC_CTX *ctx)

Referenced by deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_hmac_digest(), ecies_decrypt(), ecies_encrypt(), hmac_digest(), and stacie_derive_seed().

5.413.3.192 void(* HMAC_CTX_init_d)(HMAC_CTX *ctx)

Referenced by deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_hmac_digest(), ecies_decrypt(), ecies_encrypt(), hmac_digest(), and stacie_derive_seed().

5.413.3.193 `int(* HMAC_Final_d)(HMAC_CTX *ctx, unsigned char *md, unsigned int *len)`

Referenced by deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_hmac_digest(), ecies_decrypt(), ecies_encrypt(), hmac_digest(), and stacie_derive_seed().

5.413.3.194 `int(* HMAC_Init_ex_d)(HMAC_CTX *ctx, const void *key, int len, const EVP_MD *md, ENGINE *impl)`

Referenced by deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_hmac_digest(), ecies_decrypt(), ecies_encrypt(), hmac_digest(), and stacie_derive_seed().

5.413.3.195 `int(* HMAC_Update_d)(HMAC_CTX *ctx, const unsigned char *data, size_t len)`

Referenced by deprecated_ecies_decrypt(), deprecated_ecies_encrypt(), deprecated_hmac_digest(), ecies_decrypt(), ecies_encrypt(), hmac_digest(), and stacie_derive_seed().

5.413.3.196 `int(* i2d_ECDSA_SIG_d)(const ECDSA_SIG *sig, unsigned char **pp)`

Referenced by _ec_sign_data().

5.413.3.197 `int(* i2d_ECPrivateKey_d)(EC_KEY *key, unsigned char **out)`

5.413.3.198 `int(* i2d_OCSP_CERTID_d)(OCSP_CERTID *a, unsigned char **out)`

Referenced by _get_cache_ocsp_id().

5.413.3.199 `int(* i2d_OCSP_RESPONSE_d)(OCSP_RESPONSE *a, unsigned char **out)`

Referenced by _serialize_ocsp_response_cb().

5.413.3.200 `int(* i2d_X509_d)(X509 *a, unsigned char **out)`

Referenced by _get_x509_cert_sha_hash().

5.413.3.201 `int(* i2o_ECPublicKey_d)(EC_KEY *key, unsigned char **out)`

5.413.3.202 `const char*(* jansson_version_d)(void)`

Jansson.

Referenced by lib_version_jansson().

5.413.3.203 `const char*(* jpeg_version_d)(void)`

JPEG.

Referenced by lib_version_jpeg().

5.413.3.204 `int(* json_array_append_d)(json_t *array, json_t *value)`

5.413.3.205 `int(* json_array_append_new_d)(json_t *array, json_t *value)`

Referenced by `portal_config_entry_flags()`, `portal_contact_detail_flags()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_contacts_list()`, `portal_endpoint_folders_list()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_message_attachments()`, `portal_message_flags_array()`, and `portal_message_tags_array()`.

5.413.3.206 `int(* json_array_clear_d)(json_t *array)`

5.413.3.207 `json_t>(* json_array_d)(void)`

Referenced by `portal_config_entry_flags()`, `portal_contact_detail_flags()`, `portal_contact_details()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_contacts_list()`, `portal_endpoint_folders_list()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_message_attachments()`, `portal_message_flags_array()`, and `portal_message_tags_array()`.

5.413.3.208 `int(* json_array_extend_d)(json_t *array, json_t *other)`

5.413.3.209 `json_t>(* json_array_get_d)(const json_t *array, size_t index)`

Referenced by `portal_endpoint_alert_acknowledge()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_tag()`, `portal_parse_flags()`, `portal_parse_json_str_array()`, and `portal_parse_sections()`.

5.413.3.210 `int(* json_array_insert_d)(json_t *array, size_t index, json_t *value)`

5.413.3.211 `int(* json_array_insert_new_d)(json_t *array, size_t index, json_t *value)`

5.413.3.212 `int(* json_array_remove_d)(json_t *array, size_t index)`

5.413.3.213 `int(* json_array_set_d)(json_t *array, size_t index, json_t *value)`

5.413.3.214 `int(* json_array_set_new_d)(json_t *array, size_t index, json_t *value)`

5.413.3.215 `size_t(* json_array_size_d)(const json_t *array)`

Referenced by `portal_endpoint_alert_acknowledge()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_tag()`, `portal_parse_flags()`, `portal_parse_json_str_array()`, and `portal_parse_sections()`.

5.413.3.216 `json_t(* json_copy_d)(json_t *value)`

5.413.3.217 `void(* json_decref_d)(json_t *json)`

Referenced by `api_response()`, `http_parse_context()`, `http_session_reset()`, `portal_config_collection()`, `portal_config_entry()`, `portal_config_entry_flags()`, `portal_contact_details()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_contacts_list()`, `portal_endpoint_error()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_endpoint_response()`, and `portal_message_attachments()`.

5.413.3.218 `json_t*(* json_deep_copy_d)(json_t *value)`

5.413.3.219 `void(* json_delete_d)(json_t *json)`

5.413.3.220 `int(* json_dump_file_d)(const json_t *json, const char *path, size_t flags)`

5.413.3.221 `int(* json_dumpf_d)(const json_t *json, FILE *output, size_t flags)`

5.413.3.222 `char*(* json_dumps_d)(const json_t *json, size_t flags)`

Referenced by `api_response()`, `portal_endpoint_error()`, and `portal_endpoint_response()`.

5.413.3.223 `int(* json_equal_d)(json_t *value1, json_t *value2)`

5.413.3.224 `json_t*(* json_false_d)(void)`

5.413.3.225 `json_t*(* json_incref_d)(json_t *json)`

5.413.3.226 `json_t*(* json_integer_d)(json_int_t value)`

Referenced by `portal_endpoint_folders_tags()`.

5.413.3.227 `int(* json_integer_set_d)(json_t *integer, json_int_t value)`

5.413.3.228 `json_int_t(* json_integer_value_d)(const json_t *integer)`

Referenced by `json_api_dispatch()`, `portal_endpoint()`, `portal_endpoint_alert_acknowledge()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, and `portal_endpoint_messages_tag()`.

5.413.3.229 `json_t*(* json_load_file_d)(const char *path, size_t flags, json_error_t *error)`

5.413.3.230 `json_t*(* json_loadf_d)(FILE *input, size_t flags, json_error_t *error)`

5.413.3.231 `json_t*(* json_loads_d)(const char *input, size_t flags, json_error_t *error)`

Referenced by `http_parse_context()`, `json_api_dispatch()`, and `portal_endpoint()`.

5.413.3.232 `json_t*(* json_null_d)(void)`

5.413.3.233 `double(* json_number_value_d)(const json_t *json)`

5.413.3.234 `int(* json_object_clear_d)(json_t *object)`

5.413.3.235 `json_t*(* json_object_d)(void)`

Referenced by `portal_config_collection()`, `portal_endpoint_ad()`, `portal_endpoint_attachments_add()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_load()`, `portal_meta()`, `portal_settings_changepass()`, and `portal_settings_identity()`.

5.413.3.236 `int>(* json_object_del_d)(json_t *object, const char *key)`

5.413.3.237 `json_t>(* json_object_get_d)(const json_t *object, const char *key)`

Referenced by `http_parse_context()`, `json_api_dispatch()`, `portal_endpoint()`, and `portal_endpoint_contacts_add()`.

5.413.3.238 `void>(* json_object_iter_at_d)(json_t *object, const char *key)`

5.413.3.239 `void>(* json_object_iter_d)(json_t *object)`

Referenced by `portal_endpoint_config_edit()`, `portal_endpoint_contacts_add()`, and `portal_endpoint_contacts_edit()`.

5.413.3.240 `const char>(* json_object_iter_key_d)(void *iter)`

Referenced by `portal_endpoint_config_edit()`, `portal_endpoint_contacts_add()`, and `portal_endpoint_contacts_edit()`.

5.413.3.241 `void>(* json_object_iter_next_d)(json_t *object, void *iter)`

Referenced by `portal_endpoint_config_edit()`, `portal_endpoint_contacts_add()`, and `portal_endpoint_contacts_edit()`.

5.413.3.242 `int(* json_object_iter_set_d)(json_t *object, void *iter, json_t *value)`

5.413.3.243 `int(* json_object_iter_set_new_d)(json_t *object, void *iter, json_t *value)`

5.413.3.244 `json_t(* json_object_iter_value_d)(void *iter)`

Referenced by `portal_endpoint_config_edit()`, `portal_endpoint_contacts_add()`, and `portal_endpoint_contacts_edit()`.

5.413.3.245 `int(* json_object_set_d)(json_t *object, const char *key, json_t *value)`

5.413.3.246 `int(* json_object_set_new_d)(json_t *object, const char *key, json_t *value)`

Referenced by `portal_config_collection()`, `portal_contact_details()`, `portal_endpoint_contacts_list()`, `portal_endpoint_folders_tags()`, and `portal_endpoint_messages_load()`.

5.413.3.247 `int(* json_object_set_new_nocheck_d)(json_t *object, const char *key, json_t *value)`

5.413.3.248 `int(* json_object_set_nocheck_d)(json_t *object, const char *key, json_t *value)`

5.413.3.249 `size_t(* json_object_size_d)(const json_t *object)`

Referenced by `portal_endpoint_contacts_edit()`, `portal_endpoint_folders_add()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, and `portal_validate_request()`.

5.413.3.250 `int(* json_object_update_d)(json_t *object, json_t *other)`

5.413.3.251 `json_t>(* json_pack_d)(const char *fmt,...)`

5.413.3.252 `json_t>(* json_pack_ex_d)(json_error_t *error, size_t flags, const char *fmt,...)`

Referenced by `portal_config_entry()`, `portal_contact_details()`, `portal_endpoint_ad()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_attachments_add()`, `portal_endpoint_contacts_list()`, `portal_endpoint_error()`, `portal_endpoint_folders_list()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_message_attachments()`, `portal_message_body()`, `portal_message_header()`, `portal_message_info()`, `portal_message_meta()`, `portal_message_security()`, `portal_message_server()`, `portal_message_source()`, `portal_meta()`, `portal_settings_change_pass()`, and `portal_settings_identity()`.

5.413.3.253 `json_t>(* json_real_d)(double value)`

5.413.3.254 `int(* json_real_set_d)(json_t *real, double value)`

5.413.3.255 `double(* json_real_value_d)(const json_t *real)`

5.413.3.256 `void(* json_set_alloc_funcs_d)(json_malloc_t malloc_fn, json_free_t free_fn)`

5.413.3.257 `json_t>(* json_string_d)(const char *value)`

Referenced by `portal_config_entry_flags()`, `portal_contact_detail_flags()`, `portal_endpoint_contacts_list()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_message_flags_array()`, and `portal_message_tags_array()`.

5.413.3.258 `json_t>(* json_string_nocheck_d)(const char *value)`

5.413.3.259 `int(* json_string_set_d)(json_t *string, const char *value)`

5.413.3.260 `int(* json_string_set_nocheck_d)(json_t *string, const char *value)`

5.413.3.261 `const char>(* json_string_value_d)(const json_t *string)`

Referenced by `http_parse_context()`, `json_api_dispatch()`, `portal_endpoint()`, `portal_endpoint_config_edit()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_messages_tag()`, `portal_parse_flags()`, `portal_parse_json_str_array()`, and `portal_parse_sections()`.

5.413.3.262 `json_t>(* json_true_d)(void)`

Referenced by `portal_endpoint_aliases()`.

5.413.3.263 `const char>(* json_type_string_d)(json_t *json)`

Referenced by `portal_endpoint()`, and `portal_endpoint_contacts_add()`.

5.413.3.264 `int(* json_unpack_d)(json_t *root, const char *fmt,...)`

5.413.3.265 `int(* json_unpack_ex_d)(json_t *root, json_error_t *error, size_t flags, const char *fmt,...)`

Referenced by `api_endpoint_auth()`, `api_endpoint_change_password()`, `api_endpoint_delete_user()`, `api_endpoint_register()`, `portal_endpoint_alert_acknowledge()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_auth()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_`

endpoint_contacts_load(), portal_endpoint_contacts_move(), portal_endpoint_contacts_remove(), portal_endpoint_folders_add(), portal_endpoint_folders_list(), portal_endpoint_folders_remove(), portal_endpoint_folders_rename(), portal_endpoint_folders_tags(), portal_endpoint_messages_copy(), portal_endpoint_messages_flag(), portal_endpoint_messages_list(), portal_endpoint_messages_load(), portal_endpoint_messages_move(), portal_endpoint_messages_remove(), portal_endpoint_messages_send(), portal_endpoint_messages_tag(), and portal_settings_changepass().

5.413.3.266 `json_t>(* json_vpack_ex_d)(json_error_t *error, size_t flags, const char *fmt, va_list ap)`

Referenced by `api_response()`, and `portal_endpoint_response()`.

5.413.3.267 `int(* json_vunpack_ex_d)(json_t *root, json_error_t *error, size_t flags, const char *fmt, va_list ap)`

5.413.3.268 `int(* lt_dlexit_d)(void)`

5.413.3.269 `int(* lzo1x_1_compress_d)(const lzo_byte *src, lzo_uint src_len, lzo_byte *dst, lzo_uintp dst_len, lzo_voidp wrkmem)`

Referenced by `compress_lzo()`.

5.413.3.270 `int(* lzo1x_decompress_safe_d)(const lzo_byte *src, lzo_uint src_len, lzo_byte *dst, lzo_uintp dst_len, lzo_voidp wrkmem)`

Referenced by `decompress_block_lzo()`, and `decompress_lzo()`.

5.413.3.271 `lzo_uint32(* lzo_adler32_d)(lzo_uint32 _adler, const lzo_bytew _buf, lzo_uint _len)`

5.413.3.272 `const char>(* lzo_version_string_d)(void)`

LZO.

Referenced by `lib_version_lzo()`.

5.413.3.273 `memcached_return_t(* memcached_add_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`

Referenced by `cache_add()`, `cache_append()`, and `cache_silent_add()`.

5.413.3.274 `memcached_return_t(* memcached_append_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`

Referenced by `cache_append()`.

5.413.3.275 `memcached_return_t(* memcached_behavior_set_d)(memcached_st *ptr, const memcached_behavior_t flag, uint64_t data)`

Referenced by `cache_start()`.

5.413.3.276 `memcached_return_t(* memcached_cas_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags, uint64_t cas)`

5.413.3.277 `memcached_st(* memcached_create_d)(memcached_st *ptr)`

Referenced by `cache_start()`.

5.413.3.278 `memcached_return_t(* memcached_decrement_d)(memcached_st *ptr, const char *key, size_t key_length, uint32_t offset, uint64_t *value)`

5.413.3.279 `memcached_return_t(* memcached_decrement_with_initial_d)(memcached_st *ptr, const char *key, size_t key_length, uint64_t offset, uint64_t initial, time_t expiration, uint64_t *value)`

Referenced by `cache_decrement()`.

5.413.3.280 `memcached_return_t(* memcached_delete_d)(memcached_st *ptr, const char *key, size_t key_length, time_t expiration)`

Referenced by `cache_delete()`.

5.413.3.281 `memcached_return_t(* memcached_flush_d)(memcached_st *ptr, time_t expiration)`

MEMCACHED.

Referenced by `cache_flush()`.

5.413.3.282 `void(* memcached_free_d)(memcached_st *ptr)`

Referenced by `cache_start()`, and `cache_stop()`.

5.413.3.283 `char*(* memcached_get_d)(memcached_st *ptr, const char *key, size_t key_length, size_t *value_length, uint32_t *flags, memcached_return_t *error)`

Referenced by `cache_get()`, and `cache_get_u64()`.

5.413.3.284 `memcached_return_t(* memcached_increment_d)(memcached_st *ptr, const char *key, size_t key_length, uint32_t offset, uint64_t *value)`

5.413.3.285 `memcached_return_t(* memcached_increment_with_initial_d)(memcached_st *ptr, const char *key, size_t key_length, uint64_t offset, uint64_t initial, time_t expiration, uint64_t *value)`

Referenced by `cache_increment()`.

5.413.3.286 `const char*(* memcached_lib_version_d)(void)`

Referenced by `lib_version_cache()`.

5.413.3.287 `memcached_return_t(* memcached_prepend_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`

5.413.3.288 `memcached_return_t(* memcached_replace_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`

5.413.3.289 `memcached_return_t(* memcached_server_add_with_weight_d)(memcached_st *ptr, const char *hostname, in_port_t port, uint32_t weight)`

Referenced by `cache_start()`.

5.413.3.290 `memcached_return_t(* memcached_set_d)(memcached_st *ptr, const char *key, size_t key_length, const char *value, size_t value_length, time_t expiration, uint32_t flags)`

Referenced by `cache_set()`, and `cache_set_u64()`.

5.413.3.291 `const char*(* memcached_strerror_d)(const memcached_st *ptr, memcached_return_t rc)`

Referenced by `cache_add()`, `cache_append()`, `cache_decrement()`, `cache_delete()`, `cache_flush()`, `cache_get()`, `cache_get_u64()`, `cache_increment()`, `cache_set()`, `cache_set_u64()`, and `cache_start()`.

5.413.3.292 `void(* my_once_free_d)(void)`

MYSQL.

Referenced by `sql_stop()`.

5.413.3.293 `my_ulonglong(* mysql_affected_rows_d)(MYSQL *mysql)`

Referenced by `sql_write_conn()`.

5.413.3.294 `const char*(* mysql_character_set_name_d)(MYSQL *mysql)`

Referenced by `serv_charset_mysql()`.

5.413.3.295 `void(* mysql_close_d)(MYSQL *mysql)`

Referenced by `serv_charset_mysql()`, `serv_schema_mysql()`, `serv_version_mysql()`, `sql_open()`, and `sql_stop()`.

5.413.3.296 `my_bool(* mysql_embedded_d)(void)`

Referenced by `serv_type_mysql()`.

5.413.3.297 `unsigned int(* mysql_errno_d)(MYSQL *mysql)`

Referenced by `sql_errno()`, `sql_error()`, `stmt_errno()`, and `stmt_error()`.

5.413.3.298 `const char*(* mysql_error_d)(MYSQL *mysql)`

Referenced by `sql_error()`, `sql_insert_conn()`, `sql_num_rows_conn()`, `sql_query_res_conn()`, `sql_write_conn()`, and `stmt_error()`.

5.413.3.299 `unsigned long(* mysql_escape_string_d)(char *to, const char *from, unsigned long length)`

5.413.3.300 `MYSQL_FIELD*(* mysql_fetch_field_d)(MYSQL_RES *result)`

Referenced by `res_bind_create()`.

5.413.3.301 `MYSQL_ROW(* mysql_fetch_row_d)(MYSQL_RES *result)`

5.413.3.302 `void(* mysql_free_result_d)(MYSQL_RES *result)`

Referenced by `res_bind_create()`, and `sql_num_rows_conn()`.

5.413.3.303 `unsigned long(* mysql_get_client_version_d)(void)`

Referenced by `lib_version_mysql()`.

5.413.3.304 `const char*(* mysql_get_server_info_d)(MYSQL *mysql)`

Referenced by `serv_version_mysql()`.

5.413.3.305 `MYSQL*(* mysql_init_d)(MYSQL *mysql)`

Referenced by `sql_open()`.

5.413.3.306 `my_ulonglong(* mysql_insert_id_d)(MYSQL *mysql)`

Referenced by `sql_insert_conn()`.

5.413.3.307 `unsigned int(* mysql_num_fields_d)(MYSQL_RES *result)`

Referenced by `res_bind_create()`.

5.413.3.308 `my_ulonglong(* mysql_num_rows_d)(MYSQL_RES *result)`

Referenced by `sql_num_rows_conn()`.

5.413.3.309 `int(* mysql_options_d)(MYSQL *mysql, enum mysql_option option, const void *arg)`

Referenced by `sql_open()`.

5.413.3.310 `int(* mysql_ping_d)(MYSQL *mysql)`

Referenced by `sql_ping()`.

5.413.3.311 `MYSQL*(* mysql_real_connect_d)(MYSQL *mysql, const char *name, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag)`

Referenced by `sql_open()`.

5.413.3.312 `int(* mysql_real_query_d)(MYSQL *mysql, const char *query, unsigned long length)`

Referenced by `sql_insert_conn()`, `sql_num_rows_conn()`, `sql_query_conn()`, `sql_query_res_conn()`, and `sql_write_conn()`.

5.413.3.313 `void(* mysql_server_end_d)(void)`

Referenced by `sql_stop()`.

5.413.3.314 `int(* mysql_server_init_d)(int argc, char **argv, char **groups)`

Referenced by `sql_start()`.

5.413.3.315 `int(* mysql_set_character_set_d)(MYSQL *mysql, const char *csname)`

5.413.3.316 `my_ulonglong(* mysql_stmt_affected_rows_d)(MYSQL_STMT *stmt)`

Referenced by `stmt_exec_affected_conn()`.

5.413.3.317 `my_bool(* mysql_stmt_attr_set_d)(MYSQL_STMT *stmt, enum enum_stmt_attr_type attr_type, const void *attr)`

Referenced by `stmt_prepare()`.

5.413.3.318 `my_bool(* mysql_stmt_bind_param_d)(MYSQL_STMT *stmt, MYSQL_BIND *bind)`

Referenced by `stmt_bind_param()`.

5.413.3.319 `my_bool(* mysql_stmt_bind_result_d)(MYSQL_STMT *stmt, MYSQL_BIND *bind)`

Referenced by `res_bind_free()`, and `res_stmt_store()`.

5.413.3.320 `my_bool(* mysql_stmt_close_d)(MYSQL_STMT *)`

Referenced by `stmt_close()`.

5.413.3.321 `unsigned int(* mysql_stmt_errno_d)(MYSQL_STMT *stmt)`

Referenced by `mail_db_update_message_folder()`, `stmt_errno()`, and `stmt_error()`.

5.413.3.322 `const char*(* mysql_stmt_error_d)(MYSQL_STMT *stmt)`

Referenced by `mail_db_update_message_folder()`, `res_bind_create()`, `res_stmt_store()`, and `stmt_error()`.

5.413.3.323 `int(* mysql_stmt_execute_d)(MYSQL_STMT *stmt)`

Referenced by `stmt_exec_affected_conn()`, `stmt_exec_conn()`, `stmt_get_result_conn()`, and `stmt_insert_conn()`.

5.413.3.324 `int(* mysql_stmt_fetch_d)(MYSQL_STMT *stmt)`

Referenced by `res_stmt_store()`.

5.413.3.325 `my_bool(* mysql_stmt_free_result_d)(MYSQL_STMT *stmt)`

Referenced by `res_stmt_store()`.

5.413.3.326 `MYSQL_STMT*(* mysql_stmt_init_d)(MYSQL *mysql)`

Referenced by `stmt_open()`.

5.413.3.327 `my_ulonglong(* mysql_stmt_insert_id_d)(MYSQL_STMT *stmt)`

Referenced by `stmt_insert_conn()`.

5.413.3.328 `my_ulonglong(* mysql_stmt_num_rows_d)(MYSQL_STMT *stmt)`

Referenced by `res_stmt_store()`.

5.413.3.329 `int(* mysql_stmt_prepare_d)(MYSQL_STMT *stmt, const char *query, unsigned long length)`

Referenced by `stmt_prepare()`.

5.413.3.330 `my_bool(* mysql_stmt_reset_d)(MYSQL_STMT *stmt)`

Referenced by `stmt_reset()`.

5.413.3.331 `MYSQL_RES(* mysql_stmt_result_metadata_d)(MYSQL_STMT *stmt)`

Referenced by `res_bind_create()`.

5.413.3.332 `int(* mysql_stmt_store_result_d)(MYSQL_STMT *stmt)`

Referenced by `res_stmt_store()`.

5.413.3.333 `MYSQL_RES(* mysql_store_result_d)(MYSQL *mysql)`

Referenced by `sql_num_rows_conn()`, and `sql_query_res_conn()`.

5.413.3.334 `void(* mysql_thread_end_d)(void)`

Referenced by `sql_thread_stop()`.

5.413.3.335 `unsigned long(* mysql_thread_id_d)(MYSQL *mysql)`

Referenced by `sql_ping()`.

5.413.3.336 `my_bool(* mysql_thread_init_d)(void)`

Referenced by `sql_thread_start()`.

5.413.3.337 `unsigned int(* mysql_thread_safe_d)(void)`

Referenced by `sql_start()`.

5.413.3.338 `EC_KEY(* o2i_ECPrivateKey_d)(EC_KEY **key, const unsigned char **in, long len)`

Referenced by `_deserialize_ec_pubkey()`, and `encrypt_deserialize_pubkey()`.

5.413.3.339 `void(* OBJ_cleanup_d)(void)`

Referenced by `ssl_stop()`.

5.413.3.340 void(* OBJ_NAME_cleanup_d)(int type)

Referenced by `ssl_stop()`.

5.413.3.341 const char*(OBJ_nid2sn_d)(int n)

Referenced by `_crypto_init()`, `cipher_block_length()`, `cipher_id()`, `cipher_key_length()`, `cipher_vector_length()`, `deprecated_ecies_decrypt()`, `deprecated_ecies_encrypt()`, `deprecated_ecies_start()`, `digest_id()`, `digest_length_output()`, `ecies_decrypt()`, `ecies_encrypt()`, `ecies_start()`, and `encrypt_ctx_new()`.

5.413.3.342 int(* OCSF_basic_verify_d)(void *bs, struct stack_st_X509 *certs, struct x509_store_st *st, unsigned long flags)

Referenced by `_do_ocsp_validation()`.

5.413.3.343 void(* OCSF_BASICRESP_free_d)(OCSF_BASICRESP *a)

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.413.3.344 void*(OCSF_cert_to_id_d)(const EVP_MD *dgst, X509 *subject, X509 *issuer)

Referenced by `_do_ocsp_validation()`.

5.413.3.345 int(* OCSF_check_nonce_d)(void *req, void *bs)

Referenced by `_do_ocsp_validation()`.

5.413.3.346 int(* OCSF_check_validity_d)(ASN1_GENERALIZEDTIME *thisupd, ASN1_GENERALIZEDTIME *nextupd, long sec, long maxsec)

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.413.3.347 int(* OCSF_parse_url_d)(const char *url, char **phost, char **pport, char **ppath, int *pssl)

Referenced by `_do_ocsp_validation()`.

5.413.3.348 int(* OCSF_REQ_CTX_add1_header_d)(OCSF_REQ_CTX *rctx, const char *name, const char *value)

Referenced by `_do_ocsp_validation()`.

5.413.3.349 int(* OCSF_REQ_CTX_set1_req_d)(OCSF_REQ_CTX *rctx, void *req)

Referenced by `_do_ocsp_validation()`.

5.413.3.350 void*(OCSF_request_add0_id_d)(void *req, void *cid)

Referenced by `_do_ocsp_validation()`.

5.413.3.351 int(* OCSF_request_add1_nonce_d)(void *req, unsigned char *val, int len)

Referenced by `_do_ocsp_validation()`.

5.413.3.352 void(* OCSO_REQUEST_free_d)(OCSO_REQUEST *a)

Referenced by _do_ocsp_validation().

5.413.3.353 OCSO_REQUEST*(* OCSO_REQUEST_new_d)(void)

Referenced by _do_ocsp_validation().

5.413.3.354 int(* OCSO_REQUEST_print_d)(BIO *bp, void *a, unsigned long flags)

Referenced by _do_ocsp_validation().

5.413.3.355 int(* OCSO_resp_find_status_d)(void *bs, void *id, int *status, int *reason, ASN1_GENERALIZEDTIME **revtime, ASN1_GENERALIZEDTIME **thisupd, ASN1_GENERALIZEDTIME **nextupd)

Referenced by _do_ocsp_validation().

5.413.3.356 void(* OCSO_RESPONSE_free_d)(OCSO_RESPONSE *a)

Referenced by _destroy_ocsp_response_cb(), _do_ocsp_validation(), and _ocsp_response_callback().

5.413.3.357 void*(* OCSO_response_get1_basic_d)(OCSO_RESPONSE *resp)

Referenced by _do_ocsp_validation(), and _dump_ocsp_response_cb().

5.413.3.358 int(* OCSO_RESPONSE_print_d)(BIO *bp, OCSO_RESPONSE *o, unsigned long flags)

Referenced by _do_ocsp_validation(), and _ocsp_response_callback().

5.413.3.359 int(* OCSO_response_status_d)(OCSO_RESPONSE *resp)

Referenced by _do_ocsp_validation(), and _dump_ocsp_response_cb().

5.413.3.360 const char*(* OCSO_response_status_str_d)(long s)

Referenced by _do_ocsp_validation(), and _dump_ocsp_response_cb().

5.413.3.361 int(* OCSO_sendreq_nbio_d)(OCSO_RESPONSE **presp, OCSO_REQ_CTX *rctx)

Referenced by _do_ocsp_validation().

5.413.3.362 OCSO_REQ_CTX*(* OCSO_sendreq_new_d)(BIO *io, const char *path, void *req, int maxline)

Referenced by _do_ocsp_validation().

5.413.3.363 void(* OPENSSL_add_all_algorithms_noconf_d)(void)

Referenced by _crypto_init(), encrypt_ctx_new(), and ssl_start().

5.413.3.364 png_uint_32(* png_access_version_number_d)(void)

PNG.

Referenced by lib_version_png().

5.413.3.365 int(* RAND_bytes_d)(unsigned char *buf, int num)

Referenced by _generate_ed25519_keypair(), _get_random_bytes(), ed25519_randombytes_unsafe(), rand_choices(), rand_get_int16(), rand_get_int32(), rand_get_int64(), rand_get_int8(), rand_get_uint16(), rand_get_uint32(), rand_get_uint64(), rand_get_uint8(), and rand_write().

5.413.3.366 void(* RAND_cleanup_d)(void)

Referenced by rand_stop().

5.413.3.367 int(* RAND_load_file_d)(const char *filename, long max_bytes)

Referenced by rand_start().

5.413.3.368 int(* RAND_status_d)(void)

Referenced by rand_start().

5.413.3.369 void(* RSA_free_d)(RSA *r)

Referenced by _destroy_dnskey().

5.413.3.370 RSA*(* RSA_new_d)(void)

Referenced by _decode_rsa_pubkey(), and _get_rsa_dnskey().

5.413.3.371 RSA*(* RSAPublicKey_dup_d)(RSA *rsa)

Referenced by _clone_dnskey_record_cb().

5.413.3.372 int(* SHA1_Final_d)(unsigned char *md, SHA_CTX *c)

Referenced by _compute_sha_hash(), and _compute_sha_hash_multibuf().

5.413.3.373 int(* SHA1_Init_d)(SHA_CTX *c)

Referenced by _compute_sha_hash(), and _compute_sha_hash_multibuf().

5.413.3.374 int(* SHA1_Update_d)(SHA_CTX *c, const void *data, size_t len)

Referenced by _compute_sha_hash(), and _compute_sha_hash_multibuf().

5.413.3.375 `int(* SHA256_Final_d)(unsigned char *md, SHA256_CTX *c)`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.413.3.376 `int(* SHA256_Init_d)(SHA256_CTX *c)`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.413.3.377 `int(* SHA256_Update_d)(SHA256_CTX *c, const void *data, size_t len)`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.413.3.378 `unsigned char*(* SHA512_d)(const unsigned char *d, size_t n, unsigned char *md)`

5.413.3.379 `int(* SHA512_Final_d)(unsigned char *md, SHA512_CTX *c)`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.413.3.380 `int(* SHA512_Init_d)(SHA512_CTX *c)`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.413.3.381 `int(* SHA512_Update_d)(SHA512_CTX *c, const void *data, size_t len)`

Referenced by `_compute_sha_hash()`, and `_compute_sha_hash_multibuf()`.

5.413.3.382 `int(* sk_num_d)(const _STACK *)`

Referenced by `_do_ocsp_validation()`, and `_dump_ocsp_response_cb()`.

5.413.3.383 `void*(* sk_pop_d)(_STACK *st)`

Referenced by `_dump_ocsp_response_cb()`.

5.413.3.384 `void(* sk_pop_free_d)(_STACK *st, void(*func)(void *))`

Referenced by `ssl_stop()`.

5.413.3.385 `void*(* sk_value_d)(const _STACK *, int)`

Referenced by `_do_ocsp_validation()`.

5.413.3.386 `SPF_errcode_t(* SPF_dns_zone_add_str_d)(SPF_dns_server_t *spf_dns_server, const char *domain, ns_type rr_type, SPF_dns_stat_t herrno, const char *data)`

5.413.3.387 `SPF_dns_server_t*(* SPF_dns_zone_new_d)(SPF_dns_server_t *layer_below, const char *name, int debug)`

5.413.3.388 `void(* SPF_get_lib_version_d)(int *major, int *minor, int *patch)`

Referenced by `lib_load_spf()`.

5.413.3.389 void(* SPF_request_free_d)(SPF_request_t *sr)

Referenced by spf_check().

5.413.3.390 SPF_request_t>(* SPF_request_new_d)(SPF_server_t *spf_server)

Referenced by spf_check().

5.413.3.391 SPF_errcode_t(* SPF_request_query_mailfrom_d)(SPF_request_t *spf_request, SPF_response_t **spf_responsep)

Referenced by spf_check().

5.413.3.392 int(* SPF_request_set_env_from_d)(SPF_request_t *sr, const char *from)

Referenced by spf_check().

5.413.3.393 SPF_errcode_t(* SPF_request_set_helo_dom_d)(SPF_request_t *sr, const char *dom)

Referenced by spf_check().

5.413.3.394 SPF_errcode_t(* SPF_request_set_ipv4_d)(SPF_request_t *sr, struct in_addr addr)

Referenced by spf_check().

5.413.3.395 SPF_errcode_t(* SPF_request_set_ipv6_d)(SPF_request_t *sr, struct in6_addr addr)

Referenced by spf_check().

5.413.3.396 void(* SPF_response_free_d)(SPF_response_t *rp)

Referenced by spf_check().

5.413.3.397 SPF_reason_t(* SPF_response_reason_d)(SPF_response_t *rp)

Referenced by spf_check().

5.413.3.398 SPF_result_t(* SPF_response_result_d)(SPF_response_t *rp)

Referenced by spf_check().

5.413.3.399 void(* SPF_server_free_d)(SPF_server_t *sp)

SPF.

Referenced by spf_stop().

5.413.3.400 SPF_server_t(* SPF_server_new_d)(SPF_server_dnstype_t dnstype, int debug)

Referenced by spf_start().

5.413.3.401 `const char*(* SPF_strerror_d)(SPF_errcode_t spf_err)`

Referenced by `spf_check()`.

5.413.3.402 `const char*(* SPF_strreason_d)(SPF_reason_t reason)`

Referenced by `spf_check()`.

5.413.3.403 `const char*(* SPF_strresult_d)(SPF_result_t result)`

Referenced by `spf_check()`.

5.413.3.404 `int(* SSL_accept_d)(SSL *ssl)`

Referenced by `tls_server_alloc()`.

5.413.3.405 `int(* SSL_CIPHER_get_bits_d)(const SSL_CIPHER *c, int *alg_bits)`

Referenced by `tls_bits()`.

5.413.3.406 `const char*(* SSL_CIPHER_get_name_d)(const SSL_CIPHER *cipher)`

Referenced by `tls_cipher()`, and `tls_suite()`.

5.413.3.407 `char*(* SSL_CIPHER_get_version_d)(const SSL_CIPHER *cipher)`

5.413.3.408 `int(* SSL_connect_d)(SSL *ssl)`

Referenced by `_ssl_connect_host()`, `_ssl_starttls()`, and `tls_client_alloc()`.

5.413.3.409 `long(* SSL_ctrl_d)(SSL *s, int cmd, long larg, void *parg)`

Referenced by `_ocsp_response_callback()`, and `_ssl_connect_host()`.

5.413.3.410 `long(* SSL_CTX_callback_ctrl_d)(SSL_CTX *, int, void(*) (void))`

Referenced by `_ssl_connect_host()`.

5.413.3.411 `int(* SSL_CTX_check_private_key_d)(const SSL_CTX *ctx)`

Referenced by `tls_server_create()`.

5.413.3.412 `long(* SSL_CTX_ctrl_d)(SSL_CTX *ctx, int cmd, long larg, void *parg)`

Referenced by `_ssl_connect_host()`, `_ssl_get_client_context()`, `tls_client_alloc()`, and `tls_server_create()`.

5.413.3.413 `void(* SSL_CTX_free_d)(SSL_CTX *ctx)`

Referenced by `_ssl_shutdown()`, `ssl_verify_privkey()`, `tls_client_alloc()`, and `tls_server_destroy()`.

5.413.3.414 `int(* SSL_CTX_load_verify_locations_d)(SSL_CTX *ctx, const char *CAfile, const char *CApath)`

5.413.3.415 `SSL_CTX*(* SSL_CTX_new_d)(const SSL_METHOD *method)`

Referenced by `_ssl_get_client_context()`, `ssl_verify_privkey()`, `tls_client_alloc()`, and `tls_server_create()`.

5.413.3.416 `int(* SSL_CTX_set_cipher_list_d)(SSL_CTX *, const char *str)`

Referenced by `_ssl_get_client_context()`, and `tls_server_create()`.

5.413.3.417 `void(* SSL_CTX_set_tmp_dh_callback_d)(SSL_CTX *ctx, DH *(*dh)(SSL *ssl, int is_export, int keylength))`

Referenced by `tls_server_create()`.

5.413.3.418 `void(* SSL_CTX_set_tmp_ecdh_callback_d)(SSL_CTX *ctx, EC_KEY *(*ecdh)(SSL *ssl, int is_export, int keylength))`

Referenced by `tls_server_create()`.

5.413.3.419 `void(* SSL_CTX_set_verify_d)(SSL_CTX *ctx, int mode, int(*cb)(int, X509_STORE_CTX *))`

Referenced by `_ssl_get_client_context()`, `tls_client_alloc()`, and `tls_server_create()`.

5.413.3.420 `int(* SSL_CTX_use_certificate_chain_file_d)(SSL_CTX *ctx, const char *file)`

Referenced by `tls_server_create()`.

5.413.3.421 `int(* SSL_CTX_use_PrivateKey_file_d)(SSL_CTX *ctx, const char *file, int type)`

Referenced by `ssl_verify_privkey()`, and `tls_server_create()`.

5.413.3.422 `int(* SSL_do_handshake_d)(SSL *s)`

5.413.3.423 `void(* SSL_free_d)(SSL *ssl)`

Referenced by `_ssl_disconnect()`, `tls_client_alloc()`, `tls_free()`, and `tls_server_alloc()`.

5.413.3.424 `SSL_CIPHER*(* SSL_get_current_cipher_d)(const SSL *ssl)`

Referenced by `tls_bits()`, `tls_cipher()`, `tls_suite()`, and `tls_version()`.

5.413.3.425 `int(* SSL_get_error_d)(const SSL *s, int ret_code)`

Referenced by `tls_client_alloc()`, `tls_continue()`, `tls_error()`, and `tls_server_alloc()`.

5.413.3.426 `int(* SSL_get_fd_d)(const SSL *s)`

Referenced by `_ssl_disconnect()`.

5.413.3.427 `struct stack_st_X509>(* SSL_get_peer_cert_chain_d)(const SSL *s)`

Referenced by `_do_ocsp_validation()`, and `_verify_dx_certificate()`.

5.413.3.428 `X509>(* SSL_get_peer_certificate_d)(const SSL *s)`

Referenced by `_do_ocsp_validation()`, and `_verify_dx_certificate()`.

5.413.3.429 `int(* SSL_get_read_ahead_d)(const SSL *s)`

5.413.3.430 `int(* SSL_get_rfd_d)(const SSL *s)`

Referenced by `_ssl_fd_loop()`.

5.413.3.431 `int(* SSL_get_shutdown_d)(const SSL *ssl)`

Referenced by `tls_status()`.

5.413.3.432 `const char>(* SSL_get_version_d)(const SSL *s)`

Referenced by `tls_cipher()`, and `tls_version()`.

5.413.3.433 `BIO(* SSL_get_wbio_d)(const SSL *ssl)`

5.413.3.434 `int(* SSL_library_init_d)(void)`

Referenced by `_crypto_init()`, `_ssl_initialize()`, `encrypt_ctx_new()`, and `ssl_start()`.

5.413.3.435 `void(* SSL_load_error_strings_d)(void)`

Referenced by `_crypto_init()`, `_push_error_stack_openssl()`, `_ssl_initialize()`, `encrypt_ctx_new()`, and `ssl_start()`.

5.413.3.436 `SSL>(* SSL_new_d)(SSL_CTX *ctx)`

Referenced by `_ssl_connect_host()`, `_ssl_starttls()`, `tls_client_alloc()`, and `tls_server_alloc()`.

5.413.3.437 `int(* SSL_peek_d)(SSL *ssl, void *buf, int num)`

5.413.3.438 `int(* SSL_pending_d)(const SSL *ssl)`

5.413.3.439 `int(* SSL_read_d)(SSL *ssl, void *buf, int num)`

Referenced by `_sgnt_resolv_read_dmtpl_line()`, `_ssl_fd_loop()`, and `tls_read()`.

5.413.3.440 `void(* SSL_set_accept_state_d)(SSL *s)`

Referenced by `tls_server_alloc()`.

5.413.3.441 void(* SSL_set_bio_d)(SSL *ssl, BIO *rbio, BIO *wbio)

Referenced by `tls_client_alloc()`, and `tls_server_alloc()`.

5.413.3.442 void(* SSL_set_connect_state_d)(SSL *s)

Referenced by `tls_client_alloc()`.

5.413.3.443 int(* SSL_set_fd_d)(SSL *s, int fd)

Referenced by `_ssl_connect_host()`, and `_ssl_starttls()`.

5.413.3.444 void(* SSL_set_read_ahead_d)(SSL *s, int yes)

5.413.3.445 int(* SSL_shutdown_d)(SSL *ssl)

Referenced by `_ssl_disconnect()`, and `tls_free()`.

5.413.3.446 char** SSL_version_str_d

Definition at line 159 of file `symbols.c`.

Referenced by `lib_load_openssl()`.

5.413.3.447 int(* SSL_want_d)(const SSL *s)

5.413.3.448 int(* SSL_write_d)(SSL *ssl, const void *buf, int num)

Referenced by `_sgnt_resolv_dmtip_issue_command()`, `_sgnt_resolv_dmtip_write_data()`, `_ssl_fd_loop()`, and `tls_write()`.

5.413.3.449 const char*(* SSLeay_version_d)(int t)

5.413.3.450 const SSL_METHOD*(* SSLv23_client_method_d)(void)

Referenced by `_ssl_get_client_context()`, `ssl_verify_privkey()`, and `tls_client_alloc()`.

5.413.3.451 const SSL_METHOD*(* SSLv23_server_method_d)(void)

Referenced by `tls_server_create()`.

5.413.3.452 void(* tcfree_d)(void *ptr)

Referenced by `tank_load()`, `tree_cursor_next()`, and `tree_find()`.

5.413.3.453 bool(* tchdbclose_d)(TCHDB *hdb)

Referenced by `tank_close()`.

5.413.3.454 bool(* tchdbdefrag_d)(TCHDB *hdb, int64_t step)

Referenced by `tank_close()`, and `tank_maintain()`.

5.413.3.455 `void(* tchdbdel_d)(TCHDB *hdb)`

Referenced by `tank_close()`, and `tank_open()`.

5.413.3.456 `int(* tchdbecode_d)(TCHDB *hdb)`

Referenced by `tank_close()`, `tank_delete()`, `tank_load()`, `tank_open()`, and `tank_store()`.

5.413.3.457 `const char*(* tchdberrmsg_d)(int ecode)`

Referenced by `tank_close()`, `tank_delete()`, `tank_load()`, `tank_open()`, and `tank_store()`.

5.413.3.458 `uint64_t(* tchdbfsiz_d)(TCHDB *hdb)`

Referenced by `tank_size()`.

5.413.3.459 `void(* tchdbget_d)(TCHDB *hdb, const void *kbuf, int ksiz, int *sp)`

Referenced by `tank_load()`.

5.413.3.460 `TCHDB*(* tchdbnew_d)(void)`

Referenced by `tank_open()`.

5.413.3.461 `bool(* tchdbopen_d)(TCHDB *hdb, const char *path, int omode)`

Referenced by `tank_open()`.

5.413.3.462 `bool(* tchdboptimize_d)(TCHDB *hdb, int64_t bnum, int8_t apow, int8_t fpow, uint8_t opts)`

Referenced by `tank_close()`.

5.413.3.463 `bool(* tchdbout_d)(TCHDB *hdb, const void *kbuf, int ksiz)`

Referenced by `tank_delete()`, and `tank_store()`.

5.413.3.464 `const char*(* tchdbpath_d)(TCHDB *hdb)`

Referenced by `tank_close()`.

5.413.3.465 `bool(* tchdbputasync_d)(TCHDB *hdb, const void *kbuf, int ksiz, const void *vbuf, int vsiz)`

Referenced by `tank_store()`.

5.413.3.466 `uint64_t(* tchdbrnum_d)(TCHDB *hdb)`

Referenced by `tank_count()`.

5.413.3.467 `bool(* tchdbsetdfunit_d)(TCHDB *hdb, int32_t dfunit)`

5.413.3.468 `bool(* tchdbsetmutex_d)(TCHDB *hdb)`

Referenced by `tank_open()`.

5.413.3.469 `bool(* tchdbsync_d)(TCHDB *hdb)`

Referenced by `tank_close()`.

5.413.3.470 `bool(* tchdbtune_d)(TCHDB *hdb, int64_t bnum, int8_t apow, int8_t fpow, uint8_t opts)`

Referenced by `tank_open()`.

5.413.3.471 `void(* tclistdel_d)(TCLIST *list)`

Referenced by `tree_truncate()`.

5.413.3.472 `int(* tclistnum_d)(const TCLIST *list)`

Referenced by `tree_truncate()`.

5.413.3.473 `const void*(* tclistval_d)(const TCLIST *list, int index, int *sp)`

Referenced by `tree_truncate()`.

5.413.3.474 `void(* tcndbdel_d)(TCNDB *tree)`

Referenced by `tree_cursor_free()`, `tree_cursor_reset()`, and `tree_free()`.

5.413.3.475 `TCNDB*(* tcndbdup_d)(TCNDB *ndb)`

Referenced by `tree_cursor_alloc()`, and `tree_cursor_reset()`.

5.413.3.476 `TCLIST*(* tcndbfwmkeys_d)(TCNDB *ndb, const void *pbuf, int psiz, int max)`

5.413.3.477 `void*(* tcndbget3_d)(TCNDB *ndb, const void *kbuf, int ksiz, int *sp)`

Referenced by `tree_cursor_next()`, and `tree_find()`.

5.413.3.478 `void*(* tcndbget_d)(TCNDB *ndb, const void *kbuf, int ksiz, int *sp)`

5.413.3.479 `bool(* tcndbgetboth_d)(TCNDB *ndb, const void *kbuf, int ksiz, void **rkbuf, int *rksiz, void **rvbuf, int *rvsiz)`

Referenced by `tree_delete()`.

5.413.3.480 `void(* tcndbiterinit_d)(TCNDB *ndb)`

Referenced by `tree_cursor_alloc()`, and `tree_cursor_reset()`.

5.413.3.481 `char*(* tcndbiternext2_d)(TCNDB *ndb)`

Referenced by `tree_cursor_next()`.

5.413.3.482 `TCNDB*(* tcndbnew2_d)(TCCMP cmp, void *cmpop)`

Referenced by `tree_alloc()`.

5.413.3.483 `bool(* tcndbout_d)(TCNDB *ndb, const void *kbuf, int ksiz)`

Referenced by `tree_delete()`.

5.413.3.484 `bool(* tcndbputkeep_d)(TCNDB *ndb, const void *kbuf, int ksiz, const void *vbuf, int vsiz)`

Referenced by `tree_insert()`.

5.413.3.485 `uint64_t(* tcndbrnum_d)(TCNDB *ndb)`

Referenced by `tree_count()`.

5.413.3.486 `void(* tctreeclear_d)(TCTREE *tree)`

Referenced by `tree_truncate()`.

5.413.3.487 `TCLIST*(* tctreekeys_d)(const TCTREE *tree)`

Referenced by `tree_truncate()`.

5.413.3.488 `TCLIST*(* tctreevals_d)(const TCTREE *tree)`

Referenced by `tree_truncate()`.

5.413.3.489 `char** tcversion_d`

TOKYO.

Referenced by `lib_version_tokyo()`, and `STACK_OF()`.

5.413.3.490 `const SSL_METHOD*(* TLSv1_server_method_d)(void)`

5.413.3.491 `int(* uncompress_d)(Bytef *dest, uLongf *destLen, const Bytef *source, uLong sourceLen)`

Referenced by `decompress_zlib()`.

5.413.3.492 `utf8proc_category_t(* utf8proc_category_d)(utf8proc_int32_t c)`

5.413.3.493 `const char*(* utf8proc_category_string_d)(utf8proc_int32_t c)`

5.413.3.494 `const char*(* utf8proc_errmsg_d)(utf8proc_ssize_t errcode)`

Referenced by `utf8_error_string()`.

5.413.3.495 `const utf8proc_property_t>(* utf8proc_get_property_d)(utf8proc_int32_t uc)`

Referenced by `utf8_length_st()`.

5.413.3.496 `utf8proc_ssize_t(* utf8proc_iterate_d)(const utf8proc_uint8_t *str, utf8proc_ssize_t strlen, utf8proc_int32_t *codepoint_ref)`

Referenced by `utf8_length_st()`, and `utf8_valid_st()`.

5.413.3.497 `const char>(* utf8proc_version_d)(void)`

UTF8.

Referenced by `lib_version_utf8proc()`.

5.413.3.498 `int(* X509_check_host_d)(X509 *x, const char *chk, size_t chklen, unsigned int flags, char **peername)`

Referenced by `_do_x509_hostname_check()`.

5.413.3.499 `int(* X509_check_issued_d)(X509 *issuer, X509 *subject)`

Referenced by `_do_ocsp_validation()`, and `_verify_dx_certificate()`.

5.413.3.500 `void(* X509_email_free_d)(struct stack_st_OPENSSL_STRING *sk)`

Referenced by `_do_ocsp_validation()`.

5.413.3.501 `struct stack_st_OPENSSL_STRING>(* X509_get1_ocsp_d)(X509 *x)`

Referenced by `_do_ocsp_validation()`.

5.413.3.502 `int(* X509_get_ext_count_d)(X509 *x)`

5.413.3.503 `X509_EXTENSION>(* X509_get_ext_d)(X509 *x, int loc)`

5.413.3.504 `X509_NAME>(* X509_get_subject_name_d)(X509 *a)`

Referenced by `_do_x509_validation()`, `_get_cert_subject_cn()`, and `_verify_certificate_callback()`.

5.413.3.505 `X509_LOOKUP_METHOD(* X509_LOOKUP_file_d)(void)`

Referenced by `_do_x509_validation()`.

5.413.3.506 `ASN1_STRING(* X509_NAME_ENTRY_get_data_d)(X509_NAME_ENTRY *ne)`

Referenced by `_get_cert_subject_cn()`.

5.413.3.507 `X509_NAME_ENTRY(* X509_NAME_get_entry_d)(X509_NAME *name, int loc)`

Referenced by `_get_cert_subject_cn()`.

5.413.3.508 `int(* X509_NAME_get_index_by_NID_d)(X509_NAME *name, int nid, int lastpos)`

Referenced by `_get_cert_subject_cn()`.

5.413.3.509 `int(* X509_NAME_get_text_by_NID_d)(X509_NAME *name, int nid, char *buf, int len)`

5.413.3.510 `char*(X509_NAME_online_d)(X509_NAME *a, char *buf, int len)`

Referenced by `_do_x509_validation()`, and `_verify_certificate_callback()`.

5.413.3.511 `X509_LOOKUP*(X509_STORE_add_lookup_d)(X509_STORE *v, X509_LOOKUP_METHOD *m)`

Referenced by `_do_x509_validation()`.

5.413.3.512 `void(* X509_STORE_CTX_free_d)(X509_STORE_CTX *ctx)`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.413.3.513 `X509*(X509_STORE_CTX_get_current_cert_d)(X509_STORE_CTX *ctx)`

Referenced by `_verify_certificate_callback()`.

5.413.3.514 `int(* X509_STORE_CTX_get_error_d)(X509_STORE_CTX *ctx)`

Referenced by `_validate_self_signed()`, and `_verify_certificate_callback()`.

5.413.3.515 `int(* X509_STORE_CTX_get_error_depth_d)(X509_STORE_CTX *ctx)`

Referenced by `_verify_certificate_callback()`.

5.413.3.516 `int(* X509_STORE_CTX_init_d)(X509_STORE_CTX *ctx, X509_STORE *store, X509 *x509, STACK_OF(X509)*chain)`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.413.3.517 `X509_STORE_CTX*(X509_STORE_CTX_new_d)(void)`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.413.3.518 `void(* X509_STORE_CTX_set_chain_d)(struct x509_store_ctx_st *ctx, struct stack_st_X509 *sk)`

Referenced by `_do_x509_validation()`.

5.413.3.519 `void(* X509_STORE_free_d)(X509_STORE *v)`

Referenced by `_do_ocsp_validation()`, `_do_x509_validation()`, `_get_cert_store()`, and `_validate_self_signed()`.

5.413.3.520 `int(* X509_STORE_load_locations_d)(X509_STORE *ctx, const char *file, const char *path)`

Referenced by `_do_x509_validation()`, and `_get_cert_store()`.

5.413.3.521 `X509_STORE*(* X509_STORE_new_d)(void)`

Referenced by `_do_x509_validation()`, `_get_cert_store()`, and `_validate_self_signed()`.

5.413.3.522 `int(* X509_STORE_set_flags_d)(X509_STORE *ctx, unsigned long flags)`

Referenced by `_do_x509_validation()`.

5.413.3.523 `int(* X509_verify_cert_d)(X509_STORE_CTX *ctx)`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.413.3.524 `const char*(* X509_verify_cert_error_string_d)(long n)`

Referenced by `_do_x509_validation()`, and `_validate_self_signed()`.

5.413.3.525 `xmlNodePtr(* xmlAddSibling_d)(xmlNodePtr cur, xmlNodePtr elem)`

Referenced by `xml_node_add_sibling()`.

5.413.3.526 `const xmlChar*(* xmlBufferContent_d)(const xmlBufferPtr buf)`

Referenced by `xml_node_get_content_st()`.

5.413.3.527 `xmlBufferPtr(* xmlBufferCreate_d)(void)`

Referenced by `xml_node_get_content_st()`.

5.413.3.528 `void(* xmlBufferFree_d)(xmlBufferPtr buf)`

Referenced by `xml_node_get_content_st()`.

5.413.3.529 `int(* xmlBufferLength_d)(const xmlBufferPtr buf)`

Referenced by `xml_node_get_content_st()`.

5.413.3.530 `void(* xmlCleanupGlobals_d)(void)`

Referenced by `xml_stop()`.

5.413.3.531 `void(* xmlCleanupParser_d)(void)`

Referenced by `xml_stop()`.

5.413.3.532 `xmlDocPtr(* xmlCtxtReadMemory_d)(xmlParserCtxtPtr ctxt, const char *buffer, int size, const char *url, const char *encoding, int options)`

Referenced by `xml_create_doc()`.

5.413.3.533 `void(* xmlDocDumpFormatMemory_d)(xmlDocPtr cur, xmlChar **mem, int *size, int format)`

Referenced by `xml_dump_doc()`.

5.413.3.534 `xmlChar*(* xmlEncodeEntitiesReentrant_d)(xmlDocPtr doc, const xmlChar *input)`

Referenced by `xml_encode()`.

5.413.3.535 `void(* xmlFreeDoc_d)(xmlDocPtr doc)`

Referenced by `xml_free_doc()`.

5.413.3.536 `void(* xmlFreeNode_d)(xmlNodePtr cur)`

Referenced by `xml_node_free()`.

5.413.3.537 `void(* xmlFreeParserCtxt_d)(xmlParserCtxtPtr ctx)`

Referenced by `xml_free_parser_ctx()`.

5.413.3.538 `void(* xmlInitParser_d)(void)`

Referenced by `xml_start()`.

5.413.3.539 `void(* xmlMemoryDump_d)(void)`

Referenced by `xml_stop()`.

5.413.3.540 `xmlNodePtr(* xmlNewNode_d)(xmlNsPtr ns, const xmlChar *name)`

Referenced by `xml_node_new()`.

5.413.3.541 `xmlParserCtxtPtr(* xmlNewParserCtxt_d)(void)`

Referenced by `xml_create_parser_ctx()`.

5.413.3.542 `int(* xmlNodeBufGetContent_d)(xmlBufferPtr buffer, xmlNodePtr cur)`

Referenced by `xml_node_get_content_st()`.

5.413.3.543 `void(* xmlNodeSetContent_d)(xmlNodePtr cur, const xmlChar *content)`

Referenced by `xml_node_set_content()`.

5.413.3.544 char xmlParserVersion_d**

XML.

Referenced by lib_load_xml(), and STACK_OF().

5.413.3.545 xmlAttrPtr(* xmlSetProp_d)(xmlNodePtr node, const xmlChar *name, const xmlChar *value)

Referenced by xml_node_set_property().

5.413.3.546 xmlXPathObjectPtr(* xmlXPathEvalExpression_d)(const xmlChar *xpath, xmlXPathContextPtr ctx)

Referenced by xml_get_xpath_int16(), xml_get_xpath_int32(), xml_get_xpath_int64(), xml_get_xpath_int8(), xml_get_xpath_node_count(), xml_get_xpath_ns(), xml_get_xpath_st(), xml_get_xpath_uint16(), xml_get_xpath_uint32(), xml_get_xpath_uint64(), xml_get_xpath_uint8(), and xml_xpath_eval().

5.413.3.547 void(* xmlXPathFreeContext_d)(xmlXPathContextPtr ctx)

Referenced by xml_free_xpath_ctx().

5.413.3.548 void(* xmlXPathFreeObject_d)(xmlXPathObjectPtr obj)

Referenced by xml_free_xpath_obj(), xml_get_xpath_int16(), xml_get_xpath_int32(), xml_get_xpath_int64(), xml_get_xpath_int8(), xml_get_xpath_node_count(), xml_get_xpath_ns(), xml_get_xpath_st(), xml_get_xpath_uint16(), xml_get_xpath_uint32(), xml_get_xpath_uint64(), and xml_get_xpath_uint8().

5.413.3.549 xmlXPathContextPtr(* xmlXPathNewContext_d)(xmlDocPtr doc)

Referenced by xml_create_xpath_ctx().

5.413.3.550 int(* xmlXPathRegisterNs_d)(xmlXPathContextPtr ctxt, const xmlChar *prefix, const xmlChar *ns_uri)

Referenced by xml_xpath_set_namespace().

5.413.3.551 const char*(* zlibVersion_d)(void)

ZLIB.

Referenced by lib_version_zlib().

5.414 src/queries.h File Reference

Defines

- #define [SELECT_DOMAINS](#) "SELECT domain, restricted, mailboxes, wildcard, dkim, spf FROM Domains"
- #define [SELECT_CONFIG](#) "SELECT name, value FROM Host_Config LEFT JOIN Hosts ON (Host_Config.hostnum = Hosts.hostnum) WHERE application = 'magmad' AND (Host_Config.hostnum IS NULL OR Hosts.hostname = ?) ORDER BY Host_Config.hostnum ASC"
- #define [SELECT_HOST_NUMBER](#) "SELECT hostnum FROM Hosts WHERE hostname = ?"
- #define [DELETE_OBJECT](#) "DELETE FROM Objects WHERE objectnum = ? AND hostnum = ? AND tank = ? AND usernum = ?"
- #define [INSERT_OBJECT](#) "INSERT INTO Objects (usernum, hostnum, tank, size, serial, flags, 'references', timestamp) VALUES (?, ?, ?, ?, 0, ?, 0, NOW())"
- #define [SELECT_USER](#) "SELECT Dispatch.secure, locked, Users.usernum, tls, overquota FROM Users INNER JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE userid = ? AND legacy = ? AND email = 1"
- #define [SELECT_USER_AUTH](#) "SELECT Dispatch.secure, locked, Users.usernum, tls, overquota FROM Users INNER JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE userid = ? AND auth = ? AND email = 1 LIMIT 1"
- #define [SELECT_USERNUM_AUTH_LEGACY](#) "SELECT usernum FROM Users WHERE userid = ? AND legacy = ? AND email = 1"
- #define [SELECT_USERNUM_AUTH](#) "SELECT usernum FROM Users WHERE userid = ? AND auth = ? AND email = 1"
- #define [SELECT_USER_RECORD](#) "SELECT legacy, Dispatch.secure, locked, tls, overquota FROM Users INNER JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE Users.usernum = ? AND email = 1"
- #define [SELECT_USER_SALT](#) "SELECT salt FROM Users WHERE userid = ?"
- #define [SELECT_USER_STORAGE_KEYS](#) "SELECT signet, 'key' FROM 'Keys' WHERE usernum = ?"
- #define [UPDATE_USER_STORAGE_KEYS](#) "INSERT INTO 'Keys' (usernum, signet, 'key') VALUES (?, ?, ?) ON DUPLICATE KEY UPDATE signet = ?, 'key' = ?"
- #define [UPDATE_USER_QUOTA_ADD](#) "UPDATE Users SET size = size + ?, overquota = IF(size < quota, 0, 1) WHERE usernum = ?"
- #define [UPDATE_USER_QUOTA_SUBTRACT](#) "UPDATE Users SET size = size - ?, overquota = IF(size < quota, 0, 1) WHERE usernum = ?"
- #define [SELECT_MAILBOX_ALIASES](#)
- #define [SELECT_ALERTS](#) "SELECT alertnum, type, message, UNIX_TIMESTAMP(created) FROM Alerts WHERE usernum = ? AND acknowledged IS NULL"
- #define [UPDATE_ALERTS_ACKNOWLEDGE](#) "UPDATE Alerts SET acknowledged = NOW() WHERE alertnum = ? AND usernum = ? AND acknowledged IS NULL"
- #define [UPDATE_LOG_POP](#) "UPDATE Log SET lastpop = NOW(), popsessions = popsessions + 1 WHERE usernum = ?"
- #define [UPDATE_LOG_IMAP](#) "UPDATE Log SET lastmap = NOW(), mapsessions = mapsessions + 1 WHERE usernum = ?"
- #define [UPDATE_LOG_WEB](#) "UPDATE Log SET lastweb = NOW(), websessions = websessions + 1 WHERE usernum = ?"
- #define [SELECT_FOLDERS](#) "SELECT foldernum, parent, 'order', foldername FROM Folders WHERE usernum = ? AND type = ?"
- #define [INSERT_FOLDER](#) "INSERT INTO Folders (usernum, foldername, 'order', parent, type) VALUES (?, ?, ?, ?, ?)"
- #define [DELETE_FOLDER](#) "DELETE FROM Folders WHERE foldernum = ? AND usernum = ? AND type = ?"
- #define [UPDATE_FOLDER](#) "UPDATE Folders SET foldername = ?, parent = ?, 'order' = ? WHERE foldernum = ? AND usernum = ? AND type = ?"
- #define [RENAME_FOLDER](#) "UPDATE Folders SET foldername = ? WHERE foldernum = ? AND usernum = ? AND type = ?"
- #define [SELECT_MESSAGES](#) "SELECT messagenum, foldernum, server, status, size, signum, sigkey, UNIX_TIMESTAMP(created) FROM Messages WHERE usernum = ? AND visible = 1 ORDER BY messagenum ASC"
- #define [UPDATE_MESSAGE_VISIBILITY](#) "UPDATE Messages SET visible = 0 WHERE messagenum = ?"
- #define [UPDATE_MESSAGE_FLAGS_ADD](#) "UPDATE Messages SET status = (status | ?) WHERE usernum = ? AND foldernum = ? AND messagenum = ?"
- #define [UPDATE_MESSAGE_FLAGS_REMOVE](#) "UPDATE Messages SET status = ((status | ?) ^ ?) WHERE usernum = ? AND foldernum = ? AND messagenum = ?"
- #define [UPDATE_MESSAGE_FLAGS_REPLACE](#) "UPDATE Messages SET status = (((status | ?) ^ ?) | ?) WHERE usernum = ? AND foldernum = ? AND messagenum = ?"
- #define [UPDATE_MESSAGE_FOLDER](#) "UPDATE Messages SET foldernum = ? WHERE messagenum = ? AND usernum = ? AND foldernum = ?"

- #define [INSERT_MESSAGE](#) "INSERT INTO Messages (usernum, foldernum, server, status, size, signum, sigkey, created) VALUES (?, ?, ?, ?, ?, ?, ?, NOW())"
- #define [INSERT_MESSAGE_DUPLICATE](#) "INSERT INTO Messages (usernum, foldernum, server, status, size, signum, sigkey, created) VALUES (?, ?, ?, ?, ?, ?, ?, FROM_UNIXTIME(?))"
- #define [DELETE_MESSAGE](#) "DELETE FROM Messages WHERE messagenum = ? AND usernum = ?"
- #define [SELECT_ALL_MESSAGE_TAGS](#) "SELECT DISTINCT tag from Message_Tags LEFT JOIN Messages ON Message_Tags.messagenum = Messages.messagenum"
- #define [DELETE_MESSAGE_TAGS](#) "DELETE FROM Message_Tags WHERE messagenum = ?"
- #define [SELECT_MESSAGE_TAGS](#) "SELECT tag FROM Message_Tags WHERE messagenum = ?"
- #define [INSERT_MESSAGE_TAG](#) "INSERT INTO Message_Tags (messagenum, tag) VALUES (?, ?)"
- #define [DELETE_MESSAGE_TAG](#) "DELETE FROM Message_Tags WHERE messagenum = ? AND tag = ?"
- #define [SELECT_AGENTS](#) "SELECT agentnum, agent, popularity FROM Agents"
- #define [SELECT_MAILBOX_ADDRESS](#) "SELECT usernum FROM Mailboxes WHERE address = ? AND usernum = ?"
- #define [SELECT_MAILBOX_ADDRESS_ANY](#) "SELECT * FROM Mailboxes WHERE address = ?"
- #define [SELECT_AUTOREPLY](#) "SELECT message FROM Autoreplies WHERE replynum = ? AND usernum = ?"
- #define [SELECT_PATTERNS](#) "SELECT pattern FROM Patterns"
- #define [SELECT_FILTERS](#) "SELECT rulenum, location, type, action, foldernum, field, label, expression FROM Filters WHERE usernum = ? ORDER BY rulenum ASC"
- #define [SELECT_MESSAGES_ROLLOUT](#) "SELECT messagenum, size, server FROM Messages WHERE usernum = ? ORDER BY created ASC LIMIT 20"
- #define [SELECT_TRANSMITTING](#) "SELECT COUNT(*) FROM Transmitting WHERE usernum = ? AND timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"
- #define [SELECT_RECEIVING](#) "SELECT COUNT(*), SUM(subnet = ?) FROM Receiving WHERE usernum = ? AND timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"
- #define [SELECT_USERS_AUTH](#) "SELECT Users.usernum, Users.locked, Users.tls, Users.domain, Dispatch.send_size_limit, Dispatch.daily_send_limit, Dispatch.class FROM Users LEFT JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE userid = ? AND legacy = ? AND email = 1"
- #define [SMTP_SELECT_USER_AUTH](#) "SELECT Users.usernum, Users.tls, Users.domain, Dispatch.send_size_limit, Dispatch.daily_send_limit, Dispatch.class FROM Users LEFT JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE userid = ? AND auth = ? AND email = 1"
- #define [SELECT_PREFS_INBOUND](#)
- #define [INSERT_TRANSMITTING](#) "INSERT INTO Transmitting (usernum, timestamp) VALUES (?, NOW())"
- #define [INSERT_SIGNATURE](#) "INSERT INTO Signatures (usernum, cryptkey, junk, signature, created) VALUES (?, ?, ?, ?, NOW())"
- #define [INSERT_RECEIVING](#) "REPLACE INTO Receiving (usernum, subnet, timestamp) VALUES (?, ?, NOW())"
- #define [UPDATE_LOG_SENT](#) "UPDATE Log SET lastsent = NOW(), totalsent = totalsent + ? WHERE usernum = ?"
- #define [UPDATE_LOG_RECEIVED](#) "UPDATE Log SET lastreceived = NOW(), totalreceived = totalreceived + ?, totalbounces = totalbounces + ? WHERE usernum = ?"
- #define [SELECT_CONTACTS](#) "SELECT 'contactnum', 'name' FROM 'Contacts' WHERE 'usernum' = ? AND 'foldernum' = ?"
- #define [INSERT_CONTACT](#) "INSERT INTO 'Contacts' ('contactnum', 'usernum', 'foldernum', 'name', 'updated', 'created') VALUES (NULL, ?, ?, ?, NOW(), NOW())"
- #define [UPDATE_CONTACT](#) "UPDATE 'Contacts' SET 'foldernum' = IFNULL(?, 'foldernum'), 'name' = IFNULL(?, 'name'), 'updated' = NOW() WHERE 'contactnum' = ? AND 'usernum' = ? AND 'foldernum' = ?"
- #define [UPDATE_CONTACT_STAMP](#) "UPDATE 'Contacts' SET 'updated' = NOW() WHERE 'contactnum' = ? AND 'usernum' = ? AND 'foldernum' = ?"
- #define [DELETE_CONTACT](#) "DELETE 'Contacts', 'Contact_Details' FROM 'Contacts' LEFT JOIN 'Contact_Details' ON 'Contacts'. 'contactnum' = 'Contact_Details'. 'contactnum' WHERE 'Contacts'. 'contactnum' = ? AND 'Contacts'. 'usernum' = ? AND 'Contacts'. 'foldernum' = ?"
- #define [UPSERT_CONTACT_DETAIL](#) "INSERT INTO 'Contact_Details' ('contactnum', 'key', 'value', 'flags') VALUES (?, ?, ?, 0) ON DUPLICATE KEY UPDATE 'value' = VALUES('value'), 'flags' = VALUES('flags')"
- #define [SELECT_CONTACT_DETAILS](#) "SELECT 'key', 'value', 'flags' FROM 'Contact_Details' WHERE 'contactnum' = ?"
- #define [DELETE_CONTACT_DETAILS](#) "DELETE FROM 'Contact_Details' WHERE 'contactnum' = ? AND 'key' = ?"
- #define [SELECT_MESSAGE_FOLDER](#) "SELECT messagenum, UNIX_TIMESTAMP(created), signum, sigkey, status, server, size FROM Messages WHERE usernum = ? AND foldernum = ? AND visible = 1 ORDER BY messagenum ASC"
- #define [UPSERT_USER_CONFIG](#) "INSERT INTO 'User_Config' ('usernum', 'key', 'value', 'flags', 'timestamp') VALUES (?, ?, ?, ?, NOW()) ON DUPLICATE KEY UPDATE 'value' = VALUES('value'), 'flags' = VALUES('flags')"

- #define [SELECT_USER_CONFIG](#) "SELECT 'key', 'value', 'flags' FROM 'User_Config' WHERE 'usernum' = ?"
- #define [DELETE_USER_CONFIG](#) "DELETE FROM 'User_Config' WHERE 'usernum' = ? AND 'key' = ?"
- #define [FETCH_SIGNATURE](#) "SELECT Users.userid, Users.legacy, Users.usernum, Signatures.junk, Signatures.cryptkey, Signatures.signature FROM Signatures LEFT JOIN Users ON (Signatures.usernum = Users.usernum) WHERE Signatures.signum = ?"
- #define [UPDATE_SIGNATURE_FLAGS_ADD](#) "UPDATE Messages SET status = (status | ?) WHERE usernum = ? AND signum = ?"
- #define [UPDATE_SIGNATURE_FLAGS_REMOVE](#) "UPDATE Messages SET status = ((status | ?) ^ ?) WHERE usernum = ? AND signum = ?"
- #define [DELETE_SIGNATURE](#) "DELETE FROM Signatures WHERE signum = ?"
- #define [REGISTER_CHECK_USERNAME](#) "SELECT usernum FROM Users WHERE userid = ?"
- #define [REGISTER_INSERT_STACIE_USER](#) "INSERT INTO Users ('userid', 'salt', 'auth', 'bonus', 'tls', 'plan', 'advertising', 'quota', 'plan_expiration') VALUES (?, ?, ?, ?, 1, ?, ?, ?, ?)"
- #define [REGISTER_INSERT_STACIE_REALMS](#) "INSERT INTO Realms ('usernum', 'serial', 'label', 'shard', 'rotated') VALUES (?, ?, ?, ?, ?)"
- #define [REGISTER_INSERT_PROFILE](#) "INSERT INTO Profile ('usernum') VALUES (?)"
- #define [REGISTER_INSERT_FOLDER_NAME](#) "INSERT INTO Folders ('usernum', 'foldername') VALUES (?, ?)"
- #define [REGISTER_INSERT_LOG](#) "INSERT INTO Log ('usernum', 'created', 'created_ip') VALUES (?, NOW(), ?)"
- #define [REGISTER_INSERT_DISPATCH](#) "INSERT INTO Dispatch ('usernum', 'secure', 'spamfolder', 'inbox', 'send_size_limit', 'recv_size_limit', 'daily_send_limit', 'daily_recv_limit', 'daily_recv_limit_ip') VALUES (?, ?, ?, ?, ?, ?, ?, ?)"
- #define [REGISTER_INSERT_MAILBOXES](#) "INSERT INTO Mailboxes ('address', 'usernum') VALUES (?, ?)"
- #define [REGISTER_FETCH_BLOCKLIST](#) "SELECT sequence FROM Banned"
- #define [DELETE_USER](#) "DELETE FROM Users WHERE userid = ?"
- #define [STATISTICS_GET_TOTAL_USERS](#) "SELECT COUNT(*) FROM Users"
- #define [STATISTICS_GET_USERS_CHECKED_EMAIL_TODAY](#) "SELECT COUNT(*) FROM Log WHERE lastpop >= DATE_SUB(NOW(), INTERVAL 1 DAY) OR lastweb >= DATE_SUB(NOW(), INTERVAL 1 DAY)"
- #define [STATISTICS_GET_USERS_CHECKED_EMAIL_WEEK](#) "SELECT COUNT(*) FROM Log WHERE lastpop >= DATE_SUB(NOW(), INTERVAL 7 DAY) OR lastweb >= DATE_SUB(NOW(), INTERVAL 7 DAY)"
- #define [STATISTICS_GET_USERS_SENT_EMAIL_TODAY](#) "SELECT COUNT(*) FROM Log WHERE lastsent >= DATE_SUB(NOW(), INTERVAL 1 DAY)"
- #define [STATISTICS_GET_USERS_SENT_EMAIL_WEEK](#) "SELECT COUNT(*) FROM Log WHERE lastsent >= DATE_SUB(NOW(), INTERVAL 7 DAY)"
- #define [STATISTICS_GET_EMAILS_RECEIVED_TODAY](#) "SELECT COUNT(*) FROM Receiving WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"
- #define [STATISTICS_GET_EMAILS_RECEIVED_WEEK](#) "SELECT COUNT(*) FROM Receiving WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 7 DAY)"
- #define [STATISTICS_GET_EMAILS_SENT_TODAY](#) "SELECT COUNT(*) FROM Transmitting WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"
- #define [STATISTICS_GET_EMAILS_SENT_WEEK](#) "SELECT COUNT(*) FROM Transmitting WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 7 DAY)"
- #define [STATISTICS_GET_USERS_REGISTERED_TODAY](#) "SELECT COUNT(*) FROM Creation WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"
- #define [STATISTICS_GET_USERS_REGISTERED_WEEK](#) "SELECT COUNT(*) FROM Creation WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 7 DAY)"
- #define [AUTH_UPDATE_USER_LOCK](#) "UPDATE Users SET locked = ?, lock_expiration = '0000-00-00' WHERE usernum = ?"
- #define [AUTH_GET_BY_USERID](#) "SELECT usernum, userid, salt, auth, bonus, legacy, locked FROM Users WHERE userid = ?"
- #define [AUTH_GET_BY_ADDRESS](#) "SELECT Users.usernum, userid, salt, auth, bonus, legacy, locked FROM Users LEFT JOIN Mailboxes ON (Users.usernum = Mailboxes.usernum) WHERE Mailboxes.address = ?"
- #define [AUTH_UPDATE_LEGACY_TO_STACIE](#) "UPDATE Users SET salt = ?, auth = ?, bonus = ?, legacy = NULL WHERE usernum = ? AND legacy = ?"
- #define [META_FETCH_USER](#) "SELECT Users.userid, Users.auth, Users.tls, Users.overquota, Dispatch.secure FROM Users INNER JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE Users.usernum = ? AND email = 1 LIMIT 1"
- #define [META_FETCH_SHARD](#) "SELECT 'shard', 'rotated' FROM 'Realms' WHERE 'usernum' = ? AND 'serial' = ? AND 'label' = ?"
- #define [META_FETCH_MAIL_KEYS](#) "SELECT signet, 'key' FROM 'Keys' WHERE usernum = ?"
- #define [META_INSERT_SHARD](#) "INSERT INTO 'Realms' ('usernum', 'serial', 'label', 'shard', 'rotated') VALUES (?, ?, ?, ?, ?)"

- #define `META_INSERT_MAIL_KEYS` "INSERT INTO 'Keys' (usernum, signet, 'key') VALUES (?, ?, ?)"
- #define `QUERIES_INIT`
- #define `STMTS_INIT`

Variables

- `chr_t * queries []`
- struct {
 MYSQL_STMT `STMTS_INIT`
} `common`

5.414.1 Define Documentation

5.414.1.1 #define AUTH_GET_BY_ADDRESS "SELECT Users.usernum, userid, salt, auth, bonus, legacy, locked FROM Users LEFT JOIN Mailboxes ON (Users.usernum = Mailboxes.usernum) WHERE Mailboxes.address = ?"

Definition at line 155 of file queries.h.

5.414.1.2 #define AUTH_GET_BY_USERID "SELECT usernum, userid, salt, auth, bonus, legacy, locked FROM Users WHERE userid = ?"

Definition at line 154 of file queries.h.

5.414.1.3 #define AUTH_UPDATE_LEGACY_TO_STACIE "UPDATE Users SET salt = ?, auth = ?, bonus = ?, legacy = NULL WHERE usernum = ? AND legacy = ?"

Definition at line 156 of file queries.h.

5.414.1.4 #define AUTH_UPDATE_USER_LOCK "UPDATE Users SET locked = ?, lock_expiration = '0000-00-00' WHERE usernum = ?"

Definition at line 153 of file queries.h.

5.414.1.5 #define DELETE_CONTACT "DELETE 'Contacts', 'Contact_Details' FROM 'Contacts' LEFT JOIN 'Contact_Details' ON 'Contacts'.'contactnum' = 'Contact_Details'.'contactnum' WHERE 'Contacts'.'contactnum' = ? AND 'Contacts'.'usernum' = ? AND 'Contacts'.'foldernum' = ?"

Definition at line 106 of file queries.h.

5.414.1.6 #define DELETE_CONTACT_DETAILS "DELETE FROM 'Contact_Details' WHERE 'contactnum' = ? AND 'key' = ?"

Definition at line 111 of file queries.h.

5.414.1.7 #define DELETE_FOLDER "DELETE FROM Folders WHERE foldernum = ? AND usernum = ? AND type = ?"

Definition at line 52 of file queries.h.

5.414.1.8 #define DELETE_MESSAGE "DELETE FROM Messages WHERE messagenum = ? AND usernum = ?"

Definition at line 65 of file queries.h.

5.414.1.9 #define DELETE_MESSAGE_TAG "DELETE FROM Message_Tags WHERE messagenum = ? AND tag = ?"

Definition at line 72 of file queries.h.

5.414.1.10 #define DELETE_MESSAGE_TAGS "DELETE FROM Message_Tags WHERE messagenum = ?"

Definition at line 69 of file queries.h.

5.414.1.11 #define DELETE_OBJECT "DELETE FROM Objects WHERE objectnum = ? AND hostnum = ? AND tank = ? AND usernum = ?"

Definition at line 20 of file queries.h.

5.414.1.12 #define DELETE_SIGNATURE "DELETE FROM Signatures WHERE signum = ?"

Definition at line 125 of file queries.h.

5.414.1.13 #define DELETE_USER "DELETE FROM Users WHERE userid = ?"

Definition at line 137 of file queries.h.

5.414.1.14 #define DELETE_USER_CONFIG "DELETE FROM 'User_Config' WHERE 'usernum' = ? AND 'key' = ?"

Definition at line 119 of file queries.h.

5.414.1.15 #define FETCH_SIGNATURE "SELECT Users.userid, Users.legacy, Users.usernum, Signatures.junk, Signatures.cryptkey, Signatures.signature FROM Signatures LEFT JOIN Users ON (Signatures.usernum = Users.usernum) WHERE Signatures.signum = ?"

Definition at line 122 of file queries.h.

5.414.1.16 #define INSERT_CONTACT "INSERT INTO 'Contacts' ('contactnum', 'usernum', 'foldernum', 'name', 'updated', 'created') VALUES (NULL, ?, ?, ?, NOW(), NOW())"

Definition at line 103 of file queries.h.

5.414.1.17 #define INSERT_FOLDER "INSERT INTO Folders (usernum, foldernum, 'order', parent, type) VALUES (?, ?, ?, ?, ?)"

Definition at line 51 of file queries.h.

5.414.1.18 #define INSERT_MESSAGE "INSERT INTO Messages (usernum, foldernum, server, status, size, signum, sigkey, created) VALUES (?, ?, ?, ?, ?, ?, ?, NOW())"

Definition at line 63 of file queries.h.

5.414.1.19 `#define INSERT_MESSAGE_DUPLICATE "INSERT INTO Messages (usernum, foldernum, server, status, size, signum, sigkey, created) VALUES (?, ?, ?, ?, ?, ?, ?, FROM_UNIXTIME(?))"`

Definition at line 64 of file queries.h.

5.414.1.20 `#define INSERT_MESSAGE_TAG "INSERT INTO Message_Tags (messagenum, tag) VALUES (?, ?)"`

Definition at line 71 of file queries.h.

5.414.1.21 `#define INSERT_OBJECT "INSERT INTO Objects (usernum, hostnum, tank, size, serial, flags, 'references', timestamp) VALUES (?, ?, ?, ?, 0, ?, 0, NOW())"`

Definition at line 21 of file queries.h.

5.414.1.22 `#define INSERT_RECEIVING "REPLACE INTO Receiving (usernum, subnet, timestamp) VALUES (?, ?, NOW())"`

Definition at line 97 of file queries.h.

5.414.1.23 `#define INSERT_SIGNATURE "INSERT INTO Signatures (usernum, cryptkey, junk, signature, created) VALUES (?, ?, ?, ?, NOW())"`

Definition at line 96 of file queries.h.

5.414.1.24 `#define INSERT_TRANSMITTING "INSERT INTO Transmitting (usernum, timestamp) VALUES (?, NOW())"`

Definition at line 95 of file queries.h.

5.414.1.25 `#define META_FETCH_MAIL_KEYS "SELECT signet, 'key' FROM 'Keys' WHERE usernum = ?"`

Definition at line 161 of file queries.h.

5.414.1.26 `#define META_FETCH_SHARD "SELECT 'shard', 'rotated' FROM 'Realms' WHERE 'usernum' = ? AND 'serial' = ? AND 'label' = ?"`

Definition at line 160 of file queries.h.

5.414.1.27 `#define META_FETCH_USER "SELECT Users.userid, Users.auth, Users.tls, Users.overquota, Dispatch.secure FROM Users INNER JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE Users.usernum = ? AND email = 1 LIMIT 1"`

Definition at line 159 of file queries.h.

5.414.1.28 `#define META_INSERT_MAIL_KEYS "INSERT INTO 'Keys' (usernum, signet, 'key') VALUES (?, ?, ?)"`

Definition at line 163 of file queries.h.

5.414.1.29 `#define META_INSERT_SHARD "INSERT INTO 'Realms' ('usernum', 'serial', 'label', 'shard', 'rotated') VALUES (?, ?, ?, ?, ?)"`

Definition at line 162 of file queries.h.

5.414.1.39 `#define REGISTER_INSERT_STACIE_USER "INSERT INTO Users ('userid', 'salt', 'auth', 'bonus', 'tls', 'plan', 'advertising', 'quota', 'plan_expiration') VALUES (?, ?, ?, ?, 1, ?, ?, ?, ?)"`

Definition at line 129 of file queries.h.

5.414.1.40 `#define RENAME_FOLDER "UPDATE Folders SET foldername = ? WHERE foldernum = ? AND usernum = ? AND type = ?"`

Definition at line 54 of file queries.h.

5.414.1.41 `#define SELECT_AGENTS "SELECT agentnum, agent, popularity FROM Agents"`

Definition at line 75 of file queries.h.

5.414.1.42 `#define SELECT_ALERTS "SELECT alertnum, type, message, UNIX_TIMESTAMP(created) FROM Alerts WHERE usernum = ? AND acknowledged IS NULL"`

Definition at line 41 of file queries.h.

5.414.1.43 `#define SELECT_ALL_MESSAGE_TAGS "SELECT DISTINCT tag from Message_Tags LEFT JOIN Messages ON Message_Tags.messageum = Messages.messageum"`

Definition at line 68 of file queries.h.

5.414.1.44 `#define SELECT_AUTOREPLY "SELECT message FROM Autoreplies WHERE replynum = ? AND usernum = ?"`

Definition at line 80 of file queries.h.

5.414.1.45 `#define SELECT_CONFIG "SELECT name, value FROM Host_Config LEFT JOIN Hosts ON (Host_Config.hostnum = Hosts.hostnum) WHERE application = 'magmad' AND (Host_Config.hostnum IS NULL OR Hosts.hostname = ?) ORDER BY Host_Config.hostnum ASC"`

Definition at line 16 of file queries.h.

5.414.1.46 `#define SELECT_CONTACT_DETAILS "SELECT 'key', 'value', 'flags' FROM 'Contact_Details' WHERE 'contactnum' = ?"`

Definition at line 110 of file queries.h.

5.414.1.47 `#define SELECT_CONTACTS "SELECT 'contactnum', 'name' FROM 'Contacts' WHERE 'usernum' = ? AND 'foldernum' = ?"`

Definition at line 102 of file queries.h.

5.414.1.48 `#define SELECT_DOMAINS "SELECT domain, restricted, mailboxes, wildcard, dkim, spf FROM Domains"`

Definition at line 12 of file queries.h.

Referenced by sanity_check().

5.414.1.49 #define SELECT_FILTERS "SELECT rulenum, location, type, action, foldernum, field, label, expression FROM Filters WHERE usernum = ? ORDER BY rulenum ASC"

Definition at line 82 of file queries.h.

5.414.1.50 #define SELECT_FOLDERS "SELECT foldernum, parent, 'order', foldername FROM Folders WHERE usernum = ? AND type = ?"

Definition at line 50 of file queries.h.

5.414.1.51 #define SELECT_HOST_NUMBER "SELECT hostnum FROM Hosts WHERE hostname = ?"

Definition at line 17 of file queries.h.

5.414.1.52 #define SELECT_MAILBOX_ADDRESS "SELECT usernum FROM Mailboxes WHERE address = ? AND usernum = ?"

Definition at line 78 of file queries.h.

5.414.1.53 #define SELECT_MAILBOX_ADDRESS_ANY "SELECT * FROM Mailboxes WHERE address = ?"

Definition at line 79 of file queries.h.

5.414.1.54 #define SELECT_MAILBOX_ALIASES

Value:

```
"SELECT Aliases.aliasnum, Mailboxes.address, Aliases.display, Aliases.selected, U
    NIX_TIMESTAMP(Aliases.created) from Mailboxes " \
"LEFT JOIN Aliases ON Mailboxes.address = Aliases.address AND Mailboxes.usernum
    = Aliases.usernum " \
"WHERE Mailboxes.usernum = ? ORDER BY selected DESC, LOWER(Aliases.display) ASC
    , LOWER(Mailboxes.address) ASC"
```

Definition at line 36 of file queries.h.

5.414.1.55 #define SELECT_MESSAGE_FOLDER "SELECT messagenum, UNIX_TIMESTAMP(created), signum, sigkey, status, server, size FROM Messages WHERE usernum = ? AND foldernum = ? AND visible = 1 ORDER BY messagenum ASC"

Definition at line 114 of file queries.h.

5.414.1.56 #define SELECT_MESSAGE_TAGS "SELECT tag FROM Message_Tags WHERE messagenum = ?"

Definition at line 70 of file queries.h.

5.414.1.57 #define SELECT_MESSAGES "SELECT messagenum, foldernum, server, status, size, signum, sigkey, UNIX_TIMESTAMP(created) FROM Messages WHERE usernum = ? AND visible = 1 ORDER BY messagenum ASC"

Definition at line 57 of file queries.h.

5.414.1.58 #define SELECT_MESSAGES_ROLLOUT "SELECT messagenum, size, server FROM Messages WHERE usernum = ? ORDER BY created ASC LIMIT 20"

Definition at line 83 of file queries.h.

5.414.1.59 #define SELECT_PATTERNS "SELECT pattern FROM Patterns"

Definition at line 81 of file queries.h.

5.414.1.60 #define SELECT_PREFS_INBOUND

Value:

```
"SELECT Mailboxes.usernum, Users.locked, Users.size, Users.quota, Users.overquota
, " \
"Users.domain, Dispatch.secure, Dispatch.bounces, Dispatch.forwarded, Dispatch.
h.rollout, " \
"Dispatch.spam, Dispatch.spamaction, Dispatch.virus, Dispatch.virusaction, Dispatch.phish, Dispatch.phishaction, " \
"Dispatch.autoreply, Dispatch.inbox, Dispatch.recv_size_limit, Dispatch.daily
_recv_limit, Dispatch.daily_recv_limit_ip, " \
"Dispatch.greylist, Dispatch.greytime, Dispatch.spf, Dispatch.spfaction, Dispatch.dkim, Dispatch.dkimaction, Dispatch.rbl, " \
"Dispatch.rblaction, Dispatch.filters, 'Keys'.signet FROM Mailboxes LEFT JOIN
Users ON Mailboxes.usernum = Users.usernum LEFT JOIN Dispatch ON " \
"Mailboxes.usernum = Dispatch.usernum LEFT JOIN 'Keys' ON Mailboxes.usernum =
'Keys'.usernum WHERE Mailboxes.address = ?"
```

Definition at line 88 of file queries.h.

5.414.1.61 #define SELECT_RECEIVING "SELECT COUNT(*), SUM(subnet = ?) FROM Receiving WHERE usernum = ? AND timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"

Definition at line 85 of file queries.h.

5.414.1.62 #define SELECT_TRANSMITTING "SELECT COUNT(*) FROM Transmitting WHERE usernum = ? AND timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"

Definition at line 84 of file queries.h.

5.414.1.63 #define SELECT_USER "SELECT Dispatch.secure, locked, Users.usernum, tls, overquota FROM Users INNER JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE userid = ? AND legacy = ? AND email = 1"

Definition at line 24 of file queries.h.

5.414.1.64 #define SELECT_USER_AUTH "SELECT Dispatch.secure, locked, Users.usernum, tls, overquota FROM Users INNER JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE userid = ? AND auth = ? AND email = 1 LIMIT 1"

Definition at line 25 of file queries.h.

5.414.1.65 #define SELECT_USER_CONFIG "SELECT 'key', 'value', 'flags' FROM 'User_Config' WHERE 'usernum' = ?"

Definition at line 118 of file queries.h.

5.414.1.66 #define SELECT_USER_RECORD "SELECT legacy, Dispatch.secure, locked, tls, overquota FROM Users INNER JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE Users.usernum = ? AND email = 1"

Definition at line 28 of file queries.h.

5.414.1.67 #define SELECT_USER_SALT "SELECT salt FROM Users WHERE userid = ?"

Definition at line 29 of file queries.h.

5.414.1.68 #define SELECT_USER_STORAGE_KEYS "SELECT signet, 'key' FROM 'Keys' WHERE usernum = ?"

Definition at line 30 of file queries.h.

5.414.1.69 #define SELECT_USERNUM_AUTH "SELECT usernum FROM Users WHERE userid = ? AND auth = ? AND email = 1"

Definition at line 27 of file queries.h.

5.414.1.70 #define SELECT_USERNUM_AUTH_LEGACY "SELECT usernum FROM Users WHERE userid = ? AND legacy = ? AND email = 1"

Definition at line 26 of file queries.h.

5.414.1.71 #define SELECT_USERS_AUTH "SELECT Users.usernum, Users.locked, Users.tls, Users.domain, Dispatch.send_size_limit, Dispatch.daily_send_limit, Dispatch.class FROM Users LEFT JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE userid = ? AND legacy = ? AND email = 1"

Definition at line 86 of file queries.h.

5.414.1.72 #define SMTP_SELECT_USER_AUTH "SELECT Users.usernum, Users.tls, Users.domain, Dispatch.send_size_limit, Dispatch.daily_send_limit, Dispatch.class FROM Users LEFT JOIN Dispatch ON Users.usernum = Dispatch.usernum WHERE userid = ? AND auth = ? AND email = 1"

Definition at line 87 of file queries.h.

5.414.1.73 #define STATISTICS_GET_EMAILS_RECEIVED_TODAY "SELECT COUNT(*) FROM Receiving WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"

Definition at line 145 of file queries.h.

5.414.1.74 #define STATISTICS_GET_EMAILS_RECEIVED_WEEK "SELECT COUNT(*) FROM Receiving WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 7 DAY)"

Definition at line 146 of file queries.h.

5.414.1.75 #define STATISTICS_GET_EMAILS_SENT_TODAY "SELECT COUNT(*) FROM Transmitting WHERE timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"

Definition at line 147 of file queries.h.

5.414.1.76 `#define STATISTICS_GET_EMAILS_SENT_WEEK "SELECT COUNT(*) FROM Transmitting WHERE timestamp
>= DATE_SUB(NOW(), INTERVAL 7 DAY)"`

Definition at line 148 of file queries.h.

5.414.1.77 `#define STATISTICS_GET_TOTAL_USERS "SELECT COUNT(*) FROM Users"`

Definition at line 140 of file queries.h.

5.414.1.78 `#define STATISTICS_GET_USERS_CHECKED_EMAIL_TODAY "SELECT COUNT(*) FROM Log WHERE lastpop
>= DATE_SUB(NOW(), INTERVAL 1 DAY) OR lastweb >= DATE_SUB(NOW(), INTERVAL 1 DAY)"`

Definition at line 141 of file queries.h.

5.414.1.79 `#define STATISTICS_GET_USERS_CHECKED_EMAIL_WEEK "SELECT COUNT(*) FROM Log WHERE lastpop
>= DATE_SUB(NOW(), INTERVAL 7 DAY) OR lastweb >= DATE_SUB(NOW(), INTERVAL 7 DAY)"`

Definition at line 142 of file queries.h.

5.414.1.80 `#define STATISTICS_GET_USERS_REGISTERED_TODAY "SELECT COUNT(*) FROM Creation WHERE
timestamp >= DATE_SUB(NOW(), INTERVAL 1 DAY)"`

Definition at line 149 of file queries.h.

5.414.1.81 `#define STATISTICS_GET_USERS_REGISTERED_WEEK "SELECT COUNT(*) FROM Creation WHERE
timestamp >= DATE_SUB(NOW(), INTERVAL 7 DAY)"`

Definition at line 150 of file queries.h.

5.414.1.82 `#define STATISTICS_GET_USERS_SENT_EMAIL_TODAY "SELECT COUNT(*) FROM Log WHERE lastsent >=
DATE_SUB(NOW(), INTERVAL 1 DAY)"`

Definition at line 143 of file queries.h.

5.414.1.83 `#define STATISTICS_GET_USERS_SENT_EMAIL_WEEK "SELECT COUNT(*) FROM Log WHERE lastsent >=
DATE_SUB(NOW(), INTERVAL 7 DAY)"`

Definition at line 144 of file queries.h.

5.414.1.84 `#define STMTS_INIT`

Definition at line 277 of file queries.h.

5.414.1.85 `#define UPDATE_ALERTS_ACKNOWLEDGE "UPDATE Alerts SET acknowledged = NOW() WHERE alertnum = ?
AND usernum = ? AND acknowledged IS NULL"`

Definition at line 42 of file queries.h.

5.414.1.86 `#define UPDATE_CONTACT "UPDATE 'Contacts' SET 'foldernum' = IFNULL(?, 'foldernum'), 'name' = IFNULL(?, 'name'), 'updated' = NOW() WHERE 'contactnum' = ? AND 'usernum' = ? AND 'foldernum' = ?"`

Definition at line 104 of file queries.h.

5.414.1.87 `#define UPDATE_CONTACT_STAMP "UPDATE 'Contacts' SET 'updated' = NOW() WHERE 'contactnum' = ? AND 'usernum' = ? AND 'foldernum' = ?"`

Definition at line 105 of file queries.h.

5.414.1.88 `#define UPDATE_FOLDER "UPDATE Folders SET foldername = ?, parent = ?, 'order' = ? WHERE foldernum = ? AND usernum = ? AND type = ?"`

Definition at line 53 of file queries.h.

5.414.1.89 `#define UPDATE_LOG_IMAP "UPDATE Log SET lastmap = NOW(), mapsessions = mapsessions + 1 WHERE usernum = ?"`

Definition at line 46 of file queries.h.

5.414.1.90 `#define UPDATE_LOG_POP "UPDATE Log SET lastpop = NOW(), popsessions = popsessions + 1 WHERE usernum = ?"`

Definition at line 45 of file queries.h.

5.414.1.91 `#define UPDATE_LOG_RECEIVED "UPDATE Log SET lastreceived = NOW(), totalreceived = totalreceived + ?, totalbounces = totalbounces + ? WHERE usernum = ?"`

Definition at line 99 of file queries.h.

5.414.1.92 `#define UPDATE_LOG_SENT "UPDATE Log SET lastsent = NOW(), totalsent = totalsent + ? WHERE usernum = ?"`

Definition at line 98 of file queries.h.

5.414.1.93 `#define UPDATE_LOG_WEB "UPDATE Log SET lastweb = NOW(), websessions = websessions + 1 WHERE usernum = ?"`

Definition at line 47 of file queries.h.

5.414.1.94 `#define UPDATE_MESSAGE_FLAGS_ADD "UPDATE Messages SET status = (status | ?) WHERE usernum = ? AND foldernum = ? AND messagenum = ?"`

Definition at line 59 of file queries.h.

5.414.1.95 `#define UPDATE_MESSAGE_FLAGS_REMOVE "UPDATE Messages SET status = ((status | ?) ^ ?) WHERE usernum = ? AND foldernum = ? AND messagenum = ?"`

Definition at line 60 of file queries.h.

5.414.1.96 `#define UPDATE_MESSAGE_FLAGS_REPLACE "UPDATE Messages SET status = (((status | ?) ^ ?) | ?) WHERE usernum = ? AND foldernum = ? AND messagenum = ?"`

Definition at line 61 of file queries.h.

5.414.1.97 `#define UPDATE_MESSAGE_FOLDER "UPDATE Messages SET foldernum = ? WHERE messagenum = ? AND usernum = ? AND foldernum = ?"`

Definition at line 62 of file queries.h.

5.414.1.98 `#define UPDATE_MESSAGE_VISIBILITY "UPDATE Messages SET visible = 0 WHERE messagenum = ?"`

Definition at line 58 of file queries.h.

5.414.1.99 `#define UPDATE_SIGNATURE_FLAGS_ADD "UPDATE Messages SET status = (status | ?) WHERE usernum = ? AND signum = ?"`

Definition at line 123 of file queries.h.

5.414.1.100 `#define UPDATE_SIGNATURE_FLAGS_REMOVE "UPDATE Messages SET status = ((status | ?) ^ ?) WHERE usernum = ? AND signum = ?"`

Definition at line 124 of file queries.h.

5.414.1.101 `#define UPDATE_USER_QUOTA_ADD "UPDATE Users SET size = size + ?, overquota = IF(size < quota, 0, 1) WHERE usernum = ?"`

Definition at line 32 of file queries.h.

5.414.1.102 `#define UPDATE_USER_QUOTA_SUBTRACT "UPDATE Users SET size = size - ?, overquota = IF(size < quota, 0, 1) WHERE usernum = ?"`

Definition at line 33 of file queries.h.

5.414.1.103 `#define UPDATE_USER_STORAGE_KEYS "INSERT INTO 'Keys' (usernum, signet, 'key') VALUES (?, ?, ?) ON DUPLICATE KEY UPDATE signet = ?, 'key' = ?"`

Definition at line 31 of file queries.h.

5.414.1.104 `#define UPSERT_CONTACT_DETAIL "INSERT INTO 'Contact_Details' ('contactnum', 'key', 'value', 'flags') VALUES (?, ?, ?, 0) ON DUPLICATE KEY UPDATE 'value' = VALUES('value'), 'flags' = VALUES('flags')"`

Definition at line 109 of file queries.h.

5.414.1.105 `#define UPSERT_USER_CONFIG "INSERT INTO 'User_Config' ('usernum', 'key', 'value', 'flags', 'timestamp') VALUES (?, ?, ?, ?, NOW()) ON DUPLICATE KEY UPDATE 'value' = VALUES('value'), 'flags' = VALUES('flags')"`

Definition at line 117 of file queries.h.

5.414.2 Variable Documentation

5.414.2.1 `struct { ... } common`

Referenced by `naked_message_set()`.

5.414.2.2 `chr_t* queries[]`

Definition at line 10 of file `stmts.c`.

Referenced by `sanity_check()`, `stmt_rebuild()`, `stmt_start()`, and `stmt_stop()`.

5.414.2.3 `MYSQL_STMT STMTS_INIT`

Definition at line 382 of file `queries.h`.

5.415 src/servers/dmtp/parse.c File Reference

```
#include "magma.h"
```

5.416 src/servers/http/parse.c File Reference

```
#include "magma.h"
```

Functions

- [int_t http_parse_origin](#) ([stringer_t](#) *s, [placer_t](#) *output)
Get the origin of a resource from a url.
- void [http_parse_pairs](#) ([connection_t](#) *con)
Parse all the GET and POST parameters present in an http client request, and store them with the connection.
- void [http_parse_context](#) ([connection_t](#) *con, [stringer_t](#) *application, [stringer_t](#) *path)
Attempt to retrieve a connected user's associated session, by searching the cookie, the POST "session" variable, and the URL.
- void [http_parse_header](#) ([connection_t](#) *con)
Parse and process the current line of input from an http client connection as an http request header, storing data in the connection's http.headers member.
- void [http_parse_method](#) ([connection_t](#) *con)
Parse an http request and determine the request method and location.
- [placer_t get_header_value_noopt](#) ([stringer_t](#) *vstring)
Get the simple value of an http header, with any optional parameters stripped away.
- [placer_t get_header_opt](#) ([stringer_t](#) *vstring, [stringer_t](#) *optname)
Get the value of a named optional parameter from an http header value.
- [bool_t multipart_get_boundary](#) ([connection_t](#) *con, [placer_t](#) *output)
Get the boundary delimiter for a request by a connection specifying a Content-Type of multipart/form-data.

5.416.1 Function Documentation

5.416.1.1 [placer_t get_header_opt](#) ([stringer_t](#) * vstring, [stringer_t](#) * optname)

Get the value of a named optional parameter from an http header value.

Note:

Only the value after the ":" in a header field is passed to this function (in other words, the header name is omitted).

Parameters:

vstring a managed string containing the single http header line value to be parsed.

optname a managed string with the name of the optional parameter to be found.

Returns:

a placer pointing to the named optional parameter of the specified http header value.

Definition at line 336 of file parse.c.

References [pl_null\(\)](#), [pl_trim_start\(\)](#), [placer_t](#), [st_cmp_cs_eq\(\)](#), [tok_get_count_st\(\)](#), and [tok_get_st\(\)](#).

Referenced by [multipart_get_boundary\(\)](#), and [portal_upload\(\)](#).

5.416.1.2 `placer_t get_header_value_noopt (stringer_t * vstring)`

Get the simple value of an http header, with any optional parameters stripped away.

Note:

Only the value after the ":" in a header field is passed to this function (in other words, the header name is omitted).

Parameters:

vstring a managed string containing the single http header line value to be parsed.

Returns:

a placer pointing to the option-free header value of the specified input line.

Definition at line 313 of file parse.c.

References `pl_init()`, `placer_t`, `st_char_get()`, `st_length_get()`, `tok_get_count_st()`, and `tok_get_st()`.

Referenced by `http_parse_header()`, and `multipart_get_boundary()`.

5.416.1.3 `void http_parse_context (connection_t * con, stringer_t * application, stringer_t * path)`

Attempt to retrieve a connected user's associated session, by searching the cookie, the POST "session" variable, and the URL.

Parameters:

con a pointer to the connection object to be queried for a session id.

application a managed string containing the application associated with the connection's pending request.

path a managed string containing the path associated with the connection's pending request.

Returns:

This function returns no value.

TODO: Develop better logic for handling cookies. Namely add the ability to forcibly trigger the Set-Cookie entity and if necessary delete the existing cookie.

Definition at line 115 of file parse.c.

References `HTTP_METHOD_POST`, `json_decref_d`, `json_loads_d`, `json_object_get_d`, `json_string_value_d`, `log_pedantic`, `NULLER`, `pl_init()`, `placer_t`, `sess_get()`, `st_char_get()`, `st_cmp_ci_starts()`, `st_length_get()`, `tok_get_pl()`, and `tok_get_st()`.

Referenced by `json_api_dispatch()`, `portal_endpoint()`, `portal_process()`, and `portal_upload()`.

5.416.1.4 `void http_parse_header (connection_t * con)`

Parse and process the current line of input from an http client connection as an http request header, storing data in the connection's `http.headers` member.

Note:

If no more http headers can be read, control is returned by setting the connection `http.mode` to `HTTP_RESPOND`. Special actions are taken to store the "Host", "User-Agent", "Cookie", and "Connection" headers.

Parameters:

con the connection object of the http client to be read, and to store the results of the operation.

Returns:

This function returns no value.

LOW: Should we bother to throw an error if `con->http.connection` isn't `HTTP_CONNECTION_NEUTRAL`?

Definition at line 177 of file `parse.c`.

References `con_print()`, `CONTIGUOUS`, `data`, `get_header_value_noopt()`, `HEAP`, `magma_t::http`, `HTTP_CONNECTION_CLOSE`, `HTTP_CONNECTION_KEEPALIVE`, `http_data_free()`, `http_data_header_parse()`, `HTTP_ERROR_500`, `HTTP_RESPOND`, `int32_conv_bl()`, `inx_alloc()`, `inx_insert()`, `magma_t::log`, `log_info`, `lower_st()`, `M_INX_LINKED`, `M_TYPE_STRINGER`, `magma`, `MANAGED_T`, `http_data_t::name`, `PLACER`, `placer_t`, `multi_t::st`, `st_char_get()`, `st_cmp_ci_eq()`, `st_dupe_opts()`, `st_length_get()`, `st_length_int()`, `st_length_set()`, `st_search_cs()`, `multi_t::val`, and `http_data_t::value`.

Referenced by `http_process()`.

5.416.1.5 void http_parse_method (connection_t * con)

Parse an http request and determine the request method and location.

Note:

This function returns no value but sets the internal method, location, and state of the underlying connection object.

Parameters:

con the client connection making an http request.

Returns:

This function returns no value.

Definition at line 269 of file `parse.c`.

References `CONTIGUOUS`, `HEAP`, `magma_t::http`, `HTTP_METHOD_CONNECT`, `HTTP_METHOD_DELETE`, `HTTP_METHOD_GET`, `HTTP_METHOD_HEAD`, `HTTP_METHOD_OPTIONS`, `HTTP_METHOD_POST`, `HTTP_METHOD_PUT`, `HTTP_METHOD_TRACE`, `HTTP_METHOD_UNSUPPORTED`, `HTTP_PARSE_HEADER`, `magma_t::log`, `log_info`, `magma`, `MANAGED_T`, `PLACER`, `placer_t`, `st_char_get()`, `st_cmp_ci_starts()`, `st_dupe_opts()`, `st_length_int()`, `tok_get_count_st()`, and `tok_get_pl()`.

Referenced by `http_process()`.

5.416.1.6 int_t http_parse_origin (stringer_t * s, placer_t * output)

Get the origin of a resource from a url.

Note:

Usually we'll be give the value of the Origin header field to work with so we'll end up returning the entire string, but if we had to fall back and use the Referrer instead this should strip off the path portion.

Parameters:

s a managed string containing the url to be parsed.

output a pointer to a placer that will be set to point to the origin string inside the user-supplied url.

Returns:

0 on success or -1 on failure.

Definition at line 18 of file `parse.c`.

References `pl_init()`, `PLACER`, `st_cmp_ci_starts()`, `st_data_get()`, `st_empty_out()`, and `st_uchar_get()`.

Referenced by `http_response_allow_cross()`.

5.416.1.7 void http_parse_pairs (connection_t * *con*)

Parse all the GET and POST parameters present in an http client request, and store them with the connection.

Note:

All valid field parameters will be stored in the connection's http.pairs object.

Parameters:

con a pointer to the connection object generating the http requesting being processed.

Returns:

This function returns no value.

Definition at line 73 of file parse.c.

References count, data, http_data_free(), HTTP_DATA_GET, HTTP_DATA_POST, http_data_value_parse(), HTTP_ERROR_500, inx_alloc(), M_INX_LINKED, pl_init(), PLACER, placer_t, st_char_get(), st_length_get(), st_search_cs(), tok_get_count_st(), and tok_get_st().

Referenced by http_requeue(), and http_response().

5.416.1.8 bool_t multipart_get_boundary (connection_t * *con*, placer_t * *output*)

Get the boundary delimiter for a request by a connection specifying a Content-Type of multipart/form-data.

Parameters:

con a pointer to the connection object of the client making the http request.

output a pointer to the address of a managed string that will receive a copy of the value of the boundary string on success.

Returns:

true on success or false on failure.

Definition at line 373 of file parse.c.

References get_header_opt(), get_header_value_noopt(), http_data_get(), HTTP_DATA_HEADER, NULLER, pl_empty(), placer_t, and http_data_t::value.

Referenced by portal_upload().

5.417 src/servers/imap/parse.c File Reference

```
#include "magma.h"
```

Functions

- [int_t imap_get_type_ar](#) ([imap_arguments_t](#) *array, [size_t](#) element)
TODO: The logic here is just plain ugly. Specifically the logic used to read from the network and advance the buffers.
- [void](#) * [imap_get_ptr](#) ([imap_arguments_t](#) *array, [size_t](#) element)
Return the value of a specified object in an array as a generic pointer.
- [stringer_t](#) * [imap_get_st_ar](#) ([imap_arguments_t](#) *array, [size_t](#) element)
Return the value of a specified object in an array as a managed string.
- [imap_arguments_t](#) * [imap_get_ar_ar](#) ([imap_arguments_t](#) *array, [size_t](#) element)
Return the value of a specified object in an array as an imap arguments array.
- [int_t](#) [imap_parse_astring](#) ([stringer_t](#) **output, [chr_t](#) **start, [size_t](#) *length)
Extract the contents of an atomic string and advance the position of the parser stream.
- [int_t](#) [imap_parse_nstring](#) ([stringer_t](#) **output, [chr_t](#) **start, [size_t](#) *length, [chr_t](#) type)
- [int_t](#) [imap_parse_qstring](#) ([stringer_t](#) **output, [chr_t](#) **start, [size_t](#) *length)
Extract the contents of a quoted string and advance the position of the parser stream.
- [int_t](#) [imap_parse_literal](#) ([connection_t](#) *con, [stringer_t](#) **output, [chr_t](#) **start, [size_t](#) *length)
Extract the contents of a literal string and advance the position of the parser stream.
- [int_t](#) [imap_parse_array](#) ([int_t](#) recursion, [connection_t](#) *con, [imap_arguments_t](#) **array, [chr_t](#) **start, [size_t](#) *length)
Extract the contents of an array argument and advance the position of the parser stream.
- [int_t](#) [imap_parse_arguments](#) ([connection_t](#) *con, [chr_t](#) **start, [size_t](#) *length)
Parse a chunk of input into an array of IMAP arguments.
- [void](#) [imap_command_log_safe](#) ([stringer_t](#) *line)
LOW: Have the merge related kinks in the argument parsing system been resolved? If so, can this function be returned to the graveyard?
- [int_t](#) [imap_command_parser](#) ([connection_t](#) *con)
Parse an input line containing an IMAP command.

5.417.1 Function Documentation

5.417.1.1 [void](#) [imap_command_log_safe](#) ([stringer_t](#) * line)

LOW: Have the merge related kinks in the argument parsing system been resolved? If so, can this function be returned to the graveyard?

Definition at line 786 of file parse.c.

References [chr_whitespace\(\)](#), [log_info](#), [mm_dupe\(\)](#), [mm_free\(\)](#), [PLACER](#), [st_char_get\(\)](#), [st_empty_out\(\)](#), [st_length_int\(\)](#), and [st_search_ci\(\)](#).

Referenced by [imap_command_parser\(\)](#).

5.417.1.2 `int_t imap_command_parser (connection_t * con)`

Parse an input line containing an IMAP command. parse.c

Note:

This function updates the protocol-specific IMAP structure with parsed values for the session's tag, command, and arguments fields. Special handling is performed for any command that is preceded by a "UID" prefix.

Parameters:

con the IMAP client connection issuing the command.

Returns:

1 on success or < 0 on error. -1: the tag could not be read. -2: the IMAP command could not be read. -3: the arguments to the IMAP command could not be read.

Definition at line 886 of file parse.c.

References `ar_free()`, `magma_t::imap`, `imap_command_log_safe()`, `imap_parse_arguments()`, `imap_parse_astring()`, `length`, `magma_t::log`, `log_pedantic`, `magma`, `PLACER`, `st_cleanup`, `st_cmp_ci_eq()`, `st_data_get()`, `st_free()`, and `st_length_get()`.

Referenced by `imap_process()`.

5.417.1.3 `imap_arguments_t* imap_get_ar_ar (imap_arguments_t * array, size_t element)`

Return the value of a specified object in an array as an imap arguments array.

Parameters:

array a pointer to an imap arguments array.

element the zero-based index of the object in the imap arguments array to be examined.

Returns:

NULL on failure, or a pointer to the specified object's value as an imap arguments array on success.

Definition at line 100 of file parse.c.

References `ar_length_get()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_type_ar()`, and `log_pedantic`.

Referenced by `imap_fetch_body()`, `imap_id()`, `imap_parse_dataitems()`, `imap_search_messages_inner()`, and `imap_status()`.

5.417.1.4 `void* imap_get_ptr (imap_arguments_t * array, size_t element)`

Return the value of a specified object in an array as a generic pointer.

Parameters:

array a pointer to an imap arguments array.

element the zero-based index of the object in the imap arguments array to be examined.

Returns:

NULL on failure, or a pointer to the specified object's value on success.

Definition at line 46 of file parse.c.

References `ar_length_get()`, and `log_pedantic`.

Referenced by `imap_append()`, `imap_fetch_body()`, and `imap_store()`.

5.417.1.5 stringer_t* imap_get_st_ar (imap_arguments_t * array, size_t element)

Return the value of a specified object in an array as a managed string.

Parameters:

array a pointer to an imap arguments array.

element the zero-based index of the object in the imap arguments array to be examined.

Returns:

NULL on failure, or a pointer to the specified object's value as a managed string on success.

Definition at line 71 of file parse.c.

References `ar_length_get()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_type_ar()`, and `log_pedantic`.

Referenced by `imap_append()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_fetch()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_fetch_body_tag()`, `imap_flag_parse()`, `imap_id()`, `imap_list()`, `imap_login()`, `imap_lsub()`, `imap_parse_dataitems()`, `imap_rename()`, `imap_search_messages_inner()`, `imap_select()`, `imap_status()`, `imap_store()`, and `imap_subscribe()`.

5.417.1.6 int_t imap_get_type_ar (imap_arguments_t * array, size_t element)

TODO: The logic here is just plain ugly. Specifically the logic used to read from the network and advance the buffers. Get the type code for a specified object in an array.

Note:

Valid return values include `IMAP_ARGUMENT_TYPE_ARRAY`, `IMAP_ARGUMENT_TYPE_ASTRING`, `IMAP_ARGUMENT_TYPE_QSTRING`, `IMAP_ARGUMENT_TYPE_NSTRING`, and `IMAP_ARGUMENT_TYPE_LITERAL`.

Parameters:

array a pointer to an imap arguments array.

element the zero-based index of the object in the imap arguments array to be examined.

Returns:

`IMAP_ARGUMENT_TYPE_EMPTY` on failure, or the element type code of the specified object on success.

Definition at line 20 of file parse.c.

References `ar_length_get()`, `IMAP_ARGUMENT_TYPE_EMPTY`, and `log_pedantic`.

Referenced by `imap_append()`, `imap_copy()`, `imap_create()`, `imap_delete()`, `imap_examine()`, `imap_fetch()`, `imap_fetch_body()`, `imap_fetch_body_header()`, `imap_flag_parse()`, `imap_get_ar_ar()`, `imap_get_st_ar()`, `imap_id()`, `imap_list()`, `imap_login()`, `imap_lsub()`, `imap_parse_dataitems()`, `imap_rename()`, `imap_search_messages_inner()`, `imap_select()`, `imap_status()`, `imap_store()`, and `imap_subscribe()`.

5.417.1.7 int_t imap_parse_arguments (connection_t * con, chr_t ** start, size_t * length)

Parse a chunk of input into an array of IMAP arguments.

See also:

[imap_parse_qstring\(\)](#), [imap_parse_literal\(\)](#), [imap_parse_array\(\)](#), [imap_parse_astring\(\)](#)

Note:

This function scans a string for an array of arguments, performing the following logic when a particular character is encountered: `:` Parse argument as quoted string with [imap_parse_qstring\(\)](#). `{` : Parse argument as literal string with [imap_parse_literal\(\)](#). `(` or `[` : Parse argument as array with [imap_parse_astring\(\)](#). `other` : Defaults to parsing argument as atomic string with `imap_parse_atomic()`. The supplied start and length pointers will be updated to reflect the input stream if the quoted string is parsed successfully.

Parameters:

start the address of a pointer to the start of the buffer to be parsed, that will be continually updated to point to the next argument in the sequence during the parsing loop.

length a pointer to a `size_t` variable that contains the length of the string to be parsed, that will be continually updated with the input stream position during the parsing loop.

Returns:

-1 on parsing error or 1 on success.

Definition at line 709 of file `parse.c`.

References `ar_append()`, `ar_free()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `IMAP_ARGUMENT_TYPE_ASTRING`, `IMAP_ARGUMENT_TYPE_LITERAL`, `IMAP_ARGUMENT_TYPE_QSTRING`, `imap_parse_array()`, `imap_parse_ascending()`, `imap_parse_literal()`, and `imap_parse_qstring()`.

Referenced by `imap_command_parser()`.

5.417.1.8 int_t imap_parse_array (int_t recursion, connection_t * con, imap_arguments_t ** array, chr_t ** start, size_t * length)

Extract the contents of an array argument and advance the position of the parser stream.

Note:

This function expects as input a string beginning either with `'('` or `'['`.

Parameters:

length a pointer to a `size_t` variable that contains the length of the string to be parsed, and that will be updated to reflect the length of the remainder of the input string that follows the parsed literal string.

recursion d

con the client IMAP connection passing the array as input to the server.

array -

start -

length -

Returns:

Definition at line 595 of file `parse.c`.

References `ar_append()`, `ar_free()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `IMAP_ARGUMENT_TYPE_LITERAL`, `IMAP_ARGUMENT_TYPE_NSTRING`, `IMAP_ARGUMENT_TYPE_QSTRING`, `IMAP_ARRAY_RECURSION_LIMIT`, `imap_parse_array()`, `imap_parse_literal()`, `imap_parse_nstring()`, `imap_parse_qstring()`, `log_pedantic`, and `type()`.

Referenced by `imap_parse_arguments()`, and `imap_parse_array()`.

5.417.1.9 int_t imap_parse_ascending (stringer_t ** output, chr_t ** start, size_t * length)

Extract the contents of an atomic string and advance the position of the parser stream.

Note:

This function scans a string, expecting printable ASCII characters until it encounters a space, , , (, or [. If any other character is encountered before that point, an error will be returned. The supplied start and length pointers will be updated to reflect the input stream if the quoted string is parsed successfully.

Parameters:

- output** the address of a managed string that will receive a copy of the atomic string's contents on success, or NULL on failure.
- start** the address of a pointer to the start of the buffer to be parsed, that will also be updated to point to the next argument in the sequence on success.
- length** a pointer to a `size_t` variable that contains the length of the string to be parsed, and which will be updated to reflect the length of the remainder of the input string that follows the parsed atomic string.

Returns:

- 1 on general error or if an invalid character was encountered, or 1 if the supplied atomic string was valid.

Definition at line 154 of file `parse.c`.

References `log_pedantic`, and `st_import()`.

Referenced by `imap_command_parser()`, and `imap_parse_arguments()`.

5.417.1.10 int_t imap_parse_literal (connection_t * con, stringer_t ** output, chr_t ** start, size_t * length)

Extract the contents of a literal string and advance the position of the parser stream.

Note:

This function expects as input a string beginning with '{' and followed by a numerical string, an optional '+', and a closing '}'. After reading in the numerical size parameter, it then attempts to read in that many bytes of input from the network stream.

Parameters:

- con** the client IMAP connection passing the literal string as input to the server.
- output** the address of a managed string that will receive a copy of the literal string's contents on success, or NULL on failure or if it is zero length.
- start** the address of a pointer to the start of the buffer to be parsed (beginning with '{'), that will also be updated to point to the next argument in the sequence on success.
- length** a pointer to a `size_t` variable that contains the length of the string to be parsed, and that will be updated to reflect the length of the remainder of the input string that follows the parsed literal string.

Returns:

- 1 on general or parse error or if an enclosing pair of double quotes was not found, or 1 if the supplied quoted string was valid.

Definition at line 381 of file `parse.c`.

References `con_read()`, `con_read_line()`, `con_write_bl()`, `line_pl_st()`, `log_pedantic`, `mm_copy()`, `mm_move()`, `number`, `pl_empty()`, `pl_length_get()`, `pl_null()`, `st_alloc()`, `st_char_get()`, `st_free()`, `st_length_set()`, and `uint64_conv_bl()`.

Referenced by `imap_parse_arguments()`, and `imap_parse_array()`.

5.417.1.11 int_t imap_parse_nstring (stringer_t ** output, chr_t ** start, size_t * length, chr_t type)

Definition at line 197 of file `parse.c`.

References `log_error`, and `st_import()`.

Referenced by `imap_parse_array()`.

5.417.1.12 `int_t imap_parse_qstring (stringer_t ** output, chr_t ** start, size_t * length)`

Extract the contents of a quoted string and advance the position of the parser stream.

Note:

This function scans a string beginning with `''` for a terminating `''`, returning an error if or is encountered first. It is also able to handle escaped characters. The supplied start and length pointers will be updated to reflect the input stream if the quoted string is parsed successfully.

Parameters:

output the address of a managed string that will receive a copy of the quoted string's contents on success, or NULL on failure or if it is zero length.

start the address of a pointer to the start of the buffer to be parsed (beginning with `\`), that will also be updated to point to the next argument in the sequence on success.

length a pointer to a `size_t` variable that contains the length of the string to be parsed, and which will be updated to reflect the length of the remainder of the input string that follows the parsed quoted string.

Returns:

-1 on general error or if an enclosing pair of double quotes was not found, or 1 if the supplied quoted string was valid.

Definition at line 251 of file parse.c.

References `log_pedantic`, `st_alloc()`, `st_char_get()`, and `st_length_set()`.

Referenced by `imap_parse_arguments()`, and `imap_parse_array()`.

5.418 src/servers/pop/parse.c File Reference

```
#include "magma.h"
```

Functions

- `bool_t pop_num_parse (connection_t *con, uint64_t *outnum, bool_t required)`
Parse the argument to a POP3 command as an unsigned number.
- `bool_t pop_top_parse (connection_t *con, uint64_t *number, uint64_t *lines)`
Parse the arguments to a POP3 TOP command.
- `stringer_t * pop_pass_parse (connection_t *con)`
Parse a POP3 PASS command.
- `stringer_t * pop_user_parse (connection_t *con)`
Parse a POP3 USER command.

5.418.1 Function Documentation

5.418.1.1 `bool_t pop_num_parse (connection_t * con, uint64_t * outnum, bool_t required)`

Parse the argument to a POP3 command as an unsigned number. parse.c

Parameters:

- con** the connection across which the pop command was issued.
- outnum** a pointer to an unsigned 64-bit integer to receive the numerical argument of the pop command on success.
- required** specifies whether or not a parameter is required.

Returns:

true if the pop command contained a valid unsigned number or false on failure.

Definition at line 17 of file parse.c.

References `length`, `log_pedantic`, `st_data_get()`, `st_length_get()`, and `uint64_conv_bl()`.

Referenced by `pop_delete()`, `pop_list()`, `pop_retr()`, and `pop_uidl()`.

5.418.1.2 `stringer_t* pop_pass_parse (connection_t * con)`

Parse a POP3 PASS command.

Note:

This function will stop reading in password characters when an invalid character is encountered.

Parameters:

- con** the POP3 client connection that issued the PASS command.

Returns:

a managed string containing the user supplied password, or NULL on failure.

Definition at line 191 of file parse.c.

References CONTIGUOUS, length, log_error, log_pedantic, MANAGED_T, SECURE, st_alloc_opts(), st_copy_in(), st_data_get(), and st_length_get().

Referenced by pop_pass().

5.418.1.3 bool_t pop_top_parse (connection_t * con, uint64_t * number, uint64_t * lines)

Parse the arguments to a POP3 TOP command.

Parameters:

con the POP3 client connection that issued the TOP command.

number a pointer to an unsigned 64 bit integer to receive the value of the TOP command's message sequence number.

lines a pointer to an unsigned 64 bit integer to receive the value of the TOP command's line number count.

Returns:

true on success or false if an error was encountered.

Definition at line 89 of file parse.c.

References length, log_pedantic, st_data_get(), st_length_get(), and uint64_conv_bl().

Referenced by pop_top().

5.418.1.4 stringer_t* pop_user_parse (connection_t * con)

Parse a POP3 USER command.

Note:

The username can be at most 255 characters and will be returned in lowercase. This function will stop reading in username characters when an invalid character is encountered.

Parameters:

con the POP3 client connection that issued the USER command.

Returns:

a managed string containing the validated username, or NULL on failure.

Definition at line 257 of file parse.c.

References length, log_pedantic, lower_chr(), st_data_get(), st_import(), and st_length_get().

Referenced by pop_user().

5.419 src/servers/smtp/parse.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * smtp_parse_rcpt_to \(connection_t *con\)](#)
- [stringer_t * smtp_parse_auth \(stringer_t *data\)](#)

Parse the parameter specified to an SMTP AUTH PLAIN or SMTP AUTH LOGIN command.

- [stringer_t * smtp_parse_mail_from_path \(connection_t *con\)](#)
- [stringer_t * smtp_parse_helo_domain \(connection_t *con\)](#)

Parse the domain specified as the parameter to an SMTP HELO or EHLO command.

5.419.1 Function Documentation

5.419.1.1 stringer_t* smtp_parse_auth (stringer_t * data)

Parse the parameter specified to an SMTP AUTH PLAIN or SMTP AUTH LOGIN command. parse.c

Note:

This function will stop reading input when an invalid base64-encoding character is encountered.

Parameters:

con the SMTP client connection issuing the AUTH command.

Returns:

a managed string containing the domain specified by the AUTH command, or NULL on failure.

Definition at line 113 of file parse.c.

References [length](#), [log_pedantic](#), [st_empty_out\(\)](#), and [st_import\(\)](#).

Referenced by [smtp_auth_login\(\)](#), and [smtp_auth_plain\(\)](#).

5.419.1.2 stringer_t* smtp_parse_helo_domain (connection_t * con)

Parse the domain specified as the parameter to an SMTP HELO or EHLO command.

Note:

Valid domain names may only contain letters, numbers, periods and hyphens. This function will return a lowercase domain name, and stop reading input when an invalid character is encountered.

Parameters:

con the SMTP client connection issuing the command.

Returns:

a managed string containing the domain specified by the HELO command, or NULL on failure.

Definition at line 310 of file parse.c.

References magma_t::helo_length_limit, length, log_pedantic, lower_chr(), magma, pl_char_get(), pl_empty(), pl_length_get(), pl_length_int(), pl_trim_end(), magma_t::smtp, and st_import().

Referenced by smtp_ehlo(), and smtp_helo().

5.419.1.3 stringer_t* smtp_parse_mail_from_path (connection_t * con)

Extract the provided path from the command line Reverse-path = Path Forward-path = Path Path = "<" [A-d-l ":"] Mailbox ">" A-d-l = At-domain *("," A-d-l) At-domain = @ domain

Parameters:

con A connection structure which presumably contains a the MAIL FROM value inside the line buffer.

Returns:

Returns a stringer with the path that was extracted or NULL to indicate a problem.

Definition at line 168 of file parse.c.

References magma_t::address_length_limit, CONSTANT, length, log_pedantic, lower_chr(), magma, pl_char_get(), pl_empty(), pl_length_get(), pl_length_int(), pl_trim(), pl_trim_end(), PLACER, placer_t, magma_t::smtp, st_cmp_ci_starts(), st_cmp_cs_eq(), st_import(), tok_get_bl(), tok_get_count_bl(), tok_get_pl(), and uint64_conv_pl().

Referenced by smtp_mail_from().

5.419.1.4 stringer_t* smtp_parse_rcpt_to (connection_t * con)

LOW: The list of characters allowed in an email address needs to be verified against the RFC's. LOW: The parser should use different lists of valid characters for the the local and domain portions of an email address. Extract the provided path from the command line Reverse-path = Path Forward-path = Path Path = "<" [A-d-l ":"] Mailbox ">" A-d-l = At-domain *("," A-d-l) At-domain = @ domain

Parameters:

con A connection structure which presumably contains a the RCPT TO value inside the line buffer.

Returns:

Returns a stringer with the path that was extracted or NULL to indicate a problem.

Definition at line 24 of file parse.c.

References magma_t::address_length_limit, length, log_pedantic, lower_chr(), magma, pl_char_get(), pl_empty(), pl_length_get(), pl_length_int(), pl_trim_end(), magma_t::smtp, and st_import().

Referenced by smtp_rcpt_to().

5.420 src/web/portal/parse.c File Reference

```
#include "magma.h"
```

Functions

- [inx_t * portal_parse_json_str_array](#) (json_t *json, size_t *nout)
Parse a json string array into a linked list of managed strings.
- [int_t portal_parse_context](#) (stringer_t *context)
Parse the context of a requested folder.
- [int_t portal_parse_action](#) (stringer_t *action)
- [int_t portal_parse_sections](#) (json_t *array, uint32_t *sections)
Parse the json-rpc messages.load parameter "section" from an array of strings into a bitmask of section flags.

5.420.1 Function Documentation

5.420.1.1 int_t portal_parse_action (stringer_t * action)

Note:

This function is currently not called anywhere in the code and may be subject to removal.

Definition at line 123 of file parse.c.

References [PLACER](#), [PORTAL_ENDPOINT_ACTION_ADD](#), [PORTAL_ENDPOINT_ACTION_INVALID](#), [PORTAL_ENDPOINT_ACTION_LIST](#), [PORTAL_ENDPOINT_ACTION_REMOVE](#), [PORTAL_ENDPOINT_ACTION_REPLACE](#), and [st_cmp_ci_eq\(\)](#).

5.420.1.2 int_t portal_parse_context (stringer_t * context)

Parse the context of a requested folder.

Note:

If no context is specified, the "mail" context is assumed ([PORTAL_ENDPOINT_CONTEXT_MAIL](#)).

Parameters:

context a managed string containing the context (supported values are "mail", "contacts", "settings", and "help").

Returns:

[PORTAL_ENDPOINT_CONTEXT_INVALID](#) on failure, or the flag of the determined context on success.

Definition at line 92 of file parse.c.

References [PLACER](#), [PORTAL_ENDPOINT_CONTEXT_CONTACTS](#), [PORTAL_ENDPOINT_CONTEXT_HELP](#), [PORTAL_ENDPOINT_CONTEXT_INVALID](#), [PORTAL_ENDPOINT_CONTEXT_MAIL](#), [PORTAL_ENDPOINT_CONTEXT_SETTINGS](#), and [st_cmp_ci_eq\(\)](#).

Referenced by [portal_endpoint_folders_add\(\)](#), [portal_endpoint_folders_remove\(\)](#), [portal_endpoint_folders_rename\(\)](#), and [portal_endpoint_folders_tags\(\)](#).

5.420.1.3 `inx_t* portal_parse_json_str_array(json_t *json, size_t *nout)`

Parse a json string array into a linked list of managed strings. parse.c

Parameters:

json a json object containing an array of strings.

nout if not NULL, an optional pointer to a size_t to receive the item count of the json string array.

Returns:

NULL on failure, or a pointer to an inx holder containing the specified json array contents as a collection of managed strings.

Definition at line 16 of file parse.c.

References count, inx_alloc(), inx_free(), inx_insert(), inx_t, json_array_get_d, json_array_size_d, json_string_value_d, log_error, log_pedantic, M_INX_LINKED, M_TYPE_UINT64, ns_length_get(), st_free(), st_import(), multi_t::u64, and multi_t::val.

Referenced by portal_endpoint_messages_send().

5.420.1.4 `int_t portal_parse_sections(json_t *array, uint32_t *sections)`

Parse the json-rpc messages.load parameter "section" from an array of strings into a bitmask of section flags.

Parameters:

array a pointer to the json object representing the "section" string array of the json request message.

sections a pointer to an unsigned 32 bit integer that will receive the translated section flags on success.

Returns:

< 0 on error (-3 for an internal server error, -2 if an unknown flag was received, or -1 on syntax error), 0 for an empty sections array, or 1 on success.

Definition at line 159 of file parse.c.

References count, json_array_get_d, json_array_size_d, json_string_value_d, NULLER, PLACER, PORTAL_ENDPOINT_MESSAGE_ATTACHMENTS, PORTAL_ENDPOINT_MESSAGE_BODY, PORTAL_ENDPOINT_MESSAGE_HEADER, PORTAL_ENDPOINT_MESSAGE_INFO, PORTAL_ENDPOINT_MESSAGE_META, PORTAL_ENDPOINT_MESSAGE_SECURITY, PORTAL_ENDPOINT_MESSAGE_SERVER, PORTAL_ENDPOINT_MESSAGE_SOURCE, and st_cmp_ci_eq().

Referenced by portal_endpoint_messages_load().

5.421 src/servers/dmtp/session.c File Reference

```
#include "magma.h"
```

Functions

- void [dmtp_session_reset](#) ([connection_t](#) *con)
Reset an DMTP session to its initialized state.
- void [dmtp_session_destroy](#) ([connection_t](#) *con)
Destroy the data associated with an DMTP session.

5.421.1 Function Documentation

5.421.1.1 void dmtp_session_destroy (connection_t * con)

Destroy the data associated with an DMTP session.

Parameters:

con the DMTP client connection to be destroyed.

Returns:

This function returns no value.

Definition at line 25 of file session.c.

Referenced by con_destroy().

5.421.1.2 void dmtp_session_reset (connection_t * con)

Reset an DMTP session to its initialized state. session.c

Parameters:

con the DMTP client connection to be reset.

Returns:

This function returns no value.

Definition at line 15 of file session.c.

Referenced by dmtp_rset().

5.422 src/servers/smtp/session.c File Reference

```
#include "magma.h"
```

Functions

- void `smtp_session_reset` (`connection_t` *con)
Reset an SMTP session to its initialized state.
- void `smtp_session_destroy` (`connection_t` *con)
Destroy the data associated with an SMTP session.
- `bool_t` `smtp_check_duplicate_recipient` (`connection_t` *con, `uint64_t` usernum)
Check whether the usernum on the inbound structure has already been used.
- `bool_t` `smtp_add_recipient` (`connection_t` *con, `stringer_t` *address)
Add an entry to an SMTP session's recipients list for outbound/relayed mail.
- void `smtp_add_inbound` (`connection_t` *con, `smtp_inbound_prefs_t` *inbound)
Add an entry to an SMTP session's inbound preferences list for local delivery.
- void `smtp_add_outbound` (`connection_t` *con, `smtp_outbound_prefs_t` *outbound)
Attach a set of SMTP outbound mail preferences to an SMTP client connection.
- void `smtp_free_recipients` (`smtp_recipients_t` *recipients)
Free a list of SMTP recipients and its underlying data.
- void `smtp_free_outbound` (`smtp_outbound_prefs_t` *outbound)
Free a set of SMTP outbound mail preferences and its underlying data.
- void `smtp_free_inbound` (`smtp_inbound_prefs_t` *inbound)
Free a list of SMTP inbound mail preferences and its underlying data.
- void `smtp_list_free_filter` (`smtp_inbound_filter_t` *filter)
Free an SMTP inbound filter and its underlying data.

5.422.1 Function Documentation

5.422.1.1 void smtp_add_inbound (connection_t * con, smtp_inbound_prefs_t * inbound)

Add an entry to an SMTP session's inbound preferences list for local delivery. session.c

Parameters:

- con** the SMTP client connection specifying the added local recipient.
inbound a pointer to the SMTP inbound preferences object to be added to the SMTP session's inbound preferences list.

Returns:

true if the requested inbound preferences object was added successfully to the inbound preferences list, or false on failure.

Definition at line 150 of file session.c.

References `smtp_inbound_prefs_t::next`.

Referenced by `smtp_rcpt_to()`.

5.422.1.2 void smtp_add_outbound (connection_t * con, smtp_outbound_prefs_t * outbound)

Attach a set of SMTP outbound mail preferences to an SMTP client connection.

Parameters:

con the SMTP client connection to which the outbound mail preferences should be attached.

outbound a pointer to the SMTP outbound mail preferences to be set for the specified connection.

Returns:

This function returns no value.

Definition at line 179 of file session.c.

Referenced by smtp_auth_login(), and smtp_auth_plain().

5.422.1.3 bool_t smtp_add_recipient (connection_t * con, stringer_t * address)

Add an entry to an SMTP session's recipients list for outbound/relayed mail.

Note:

If the recipients list does not exist, it will be created automatically.

Parameters:

con the SMTP client connection specifying the added recipient.

address a managed string containing the recipient's email address.

Returns:

true if the requested recipient was added successfully to the recipients list, or false on failure.

Definition at line 107 of file session.c.

References smtp_recipients_t::address, log_pedantic, mm_alloc(), mm_free(), smtp_recipients_t::next, and st_dupe().

Referenced by smtp_rcpt_to().

5.422.1.4 bool_t smtp_check_duplicate_recipient (connection_t * con, uint64_t usernum)

Check whether the usernum on the inbound structure has already been used.

Definition at line 80 of file session.c.

References smtp_inbound_prefs_t::next, and smtp_inbound_prefs_t::usernum.

Referenced by smtp_rcpt_to().

5.422.1.5 void smtp_free_inbound (smtp_inbound_prefs_t * inbound)

Free a list of SMTP inbound mail preferences and its underlying data.

Parameters:

inbound a pointer to the head of the SMTP inbound mail preferences list to be destroyed.

Returns:

This function returns no value.

Definition at line 228 of file session.c.

References `smtp_inbound_prefs_t::address`, `smtp_inbound_prefs_t::domain`, `smtp_inbound_prefs_t::filters`, `smtp_inbound_prefs_t::forwarded`, `inx_cleanup()`, `mm_free()`, `smtp_inbound_prefs_t::next`, `prime_cleanup()`, `smtp_inbound_prefs_t::rcptto`, `smtp_inbound_prefs_t::signet`, `smtp_inbound_prefs_t::spamsig`, and `st_cleanup`.

Referenced by `smtp_fetch_inbound()`, `smtp_rcpt_to()`, `smtp_session_destroy()`, and `smtp_session_reset()`.

5.422.1.6 void smtp_free_outbound (smtp_outbound_prefs_t * *outbound*)

Free a set of SMTP outbound mail preferences and its underlying data.

Parameters:

outbound a pointer to the SMTP outbound mail preferences set to be destroyed.

Returns:

This function returns no value.

Definition at line 210 of file session.c.

References `smtp_outbound_prefs_t::domain`, `mm_free()`, `smtp_outbound_prefs_t::recipients`, `smtp_free_recipients()`, and `st_cleanup`.

Referenced by `smtp_session_destroy()`.

5.422.1.7 void smtp_free_recipients (smtp_recipients_t * *recipients*)

Free a list of SMTP recipients and its underlying data.

Parameters:

recipients a pointer to the head of the recipients list to be destroyed.

Returns:

This function returns no value.

Definition at line 191 of file session.c.

References `smtp_recipients_t::address`, `mm_free()`, `smtp_recipients_t::next`, and `st_cleanup`.

Referenced by `smtp_free_outbound()`, and `smtp_session_reset()`.

5.422.1.8 void smtp_list_free_filter (smtp_inbound_filter_t * *filter*)

Free an SMTP inbound filter and its underlying data.

Parameters:

filter a pointer to the SMTP inbound filter to be destroyed.

Returns:

This function returns no value.

Definition at line 249 of file session.c.

References `smtp_inbound_filter_t::expression`, `smtp_inbound_filter_t::field`, `smtp_inbound_filter_t::label`, `mm_free()`, and `st_cleanup`.

Referenced by `smtp_fetch_inbound()`.

5.422.1.9 void smtp_session_destroy (connection_t * con)

Destroy the data associated with an SMTP session.

Parameters:

con the SMTP client connection to be destroyed.

Returns:

This function returns no value.

Definition at line 57 of file session.c.

References mail_destroy_message(), smtp_free_inbound(), smtp_free_outbound(), and st_cleanup.

Referenced by con_destroy().

5.422.1.10 void smtp_session_reset (connection_t * con)

Reset an SMTP session to its initialized state.

Parameters:

con the SMTP client connection to be reset.

Returns:

This function returns no value.

Definition at line 15 of file session.c.

References mail_destroy_message(), smtp_free_inbound(), smtp_free_recipients(), and st_cleanup.

Referenced by smtp_auth_login(), smtp_auth_plain(), smtp_data_inbound(), smtp_data_outbound(), smtp_mail_from(), smtp_rset(), and smtp_starttls().

5.423 src/servers/http/content.c File Reference

```
#include "magma.h"
```

Functions

- void [http_free_content](#) ([http_content_t](#) *page)
LOW: We should use `basename()` and `dirname()` to cleanup path strings.
- void [http_page_free](#) ([http_page_t](#) *page)
Free an http page object and all its underlying data.
- [http_content_t](#) * [http_get_static](#) ([stringer_t](#) *location)
Get a cached copy of a static web page.
- [http_content_t](#) * [http_get_template](#) ([chr_t](#) *location)
Return a template by location.
- [http_page_t](#) * [http_page_get](#) ([chr_t](#) *location)
Get a template page and prepare its xml document root for use.
- [bool_t](#) [http_load_file](#) ([int_t](#) template, [chr_t](#) *filename)
Load file content into the http server repository.
- [bool_t](#) [http_content_load_directory](#) ([int_t](#) template, [chr_t](#) *directory)
Load the content from a directory into the http server repository, recursively.
- [bool_t](#) [http_content_load_fonts](#) (void)
Load all fonts into the http server repository.
- [bool_t](#) [http_content_start](#) (void)
Prime the the web app templates, static content, and fonts directories for future use.
- void [http_content_stop](#) (void)
Purge the http contents repository of stored fonts, templates, and static web pages.
- [bool_t](#) [http_content_refresh](#) (void)
Refresh the stored contents of the web app templates, static content, and fonts directories.

Variables

- struct {
 [inx_t](#) * fonts
 [inx_t](#) * pages
 [inx_t](#) * templates
} [content](#)

5.423.1 Function Documentation

5.423.1.1 `bool_t http_content_load_directory (int_t template, chr_t * directory)`

Load the content from a directory into the http server repository, recursively. [content.c](#)

See also:

[http_load_file\(\)](#)

Parameters:

template a value specifying whether the specified directory contains templates or regular web content.

directory a null-terminated string specifying the pathname of the directory to be scanned recursively.

Returns:

0 on failure or 1 on success.

Definition at line 313 of file [content.c](#).

References [http_content_load_directory\(\)](#), [http_load_file\(\)](#), [log_info](#), [MEMORYBUF](#), [NULLER](#), [PLACER](#), [st_char_get\(\)](#), [st_cmp_cs_ends\(\)](#), [st_cmp_cs_eq\(\)](#), [st_free\(\)](#), and [st_merge](#).

Referenced by [http_content_load_directory\(\)](#), [http_content_refresh\(\)](#), and [http_content_start\(\)](#).

5.423.1.2 `bool_t http_content_load_fonts (void)`

Load all fonts into the http server repository.

Note:

This function will load all fonts of extension ".ttf" from the directory configured in [magma.http.fonts](#).

Returns:

0 on failure or 1 on success.

Definition at line 362 of file [content.c](#).

References [magma_t::content](#), [content](#), [magma_t::http](#), [inx_insert\(\)](#), [magma_t::log](#), [log_info](#), [log_pedantic](#), [M_TYPE_UINT64](#), [magma](#), [MEMORYBUF](#), [NULLER](#), [PLACER](#), [st_cmp_ci_ends\(\)](#), [st_cmp_cs_ends\(\)](#), [st_free\(\)](#), [st_merge](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [http_content_refresh\(\)](#), and [http_content_start\(\)](#).

5.423.1.3 `bool_t http_content_refresh (void)`

Refresh the stored contents of the web app templates, static content, and fonts directories.

Returns:

true if all content directories were successfully refreshed, or false on failure.

Definition at line 449 of file [content.c](#).

References [content](#), [magma_t::http](#), [http_content_load_directory\(\)](#), [http_content_load_fonts\(\)](#), [http_free_content\(\)](#), [inx_alloc\(\)](#), [inx_free\(\)](#), [M_INX_LINKED](#), [magma](#), and [st_free\(\)](#).

Referenced by [signal_refresh\(\)](#).

5.423.1.4 `bool_t http_content_start (void)`

Prime the the web app templates, static content, and fonts directories for future use.

Note:

This function makes sure that the web templates, static content, and fonts directory have been properly set, and loads and caches their content for future use.

Returns:

true on success or false on failure.

Definition at line 407 of file content.c.

References content, magma_t::http, http_content_load_directory(), http_content_load_fonts(), http_free_content(), inx_alloc(), log_pedantic, M_INX_LINKED, magma, NULLER, PLACER, st_cmp_cs_ends(), and st_free().

Referenced by process_start().

5.423.1.5 `void http_content_stop (void)`

Purge the http contents repository of stored fonts, templates, and static web pages.

Returns:

This function returns no value.

Definition at line 435 of file content.c.

References content, and inx_cleanup().

Referenced by process_stop().

5.423.1.6 `void http_free_content (http_content_t * page)`

LOW: We should use basename() and dirname() to cleanup path strings. LOW: These functions should all be renamed to http_content_XXXX. The file and directory functions should be updated to use the x64/reentrant alternatives. Specifically open64/fstat64/readdir64_r. An update function that can be triggered with a SIGHUP would also be nice. Free a stored piece of http content and all its underlying data.

Parameters:

page a pointer to the loaded content page to be freed.

Returns:

This function returns no value.

Definition at line 27 of file content.c.

References http_content_t::location, mm_free(), http_content_t::resource, st_cleanup, and http_content_t::type.

Referenced by http_content_refresh(), http_content_start(), and http_load_file().

5.423.1.7 `http_content_t* http_get_static (stringer_t * location)`

Get a cached copy of a static web page.

Parameters:

location a pointer to a managed string containing the location of the requested resource.

Returns:

NULL on failure, or a pointer to an http content object with the contents of the requested resource on success.

Definition at line 67 of file content.c.

References content, inx_find(), M_TYPE_STRINGER, and http_content_t::type.

Referenced by http_response().

5.423.1.8 http_content_t* http_get_template (chr_t * location)

Return a template by location.

Parameters:

location a string specifying the location of the template.

Returns:

NULL on failure, or a pointer to an [http_content_t](#) object containing the template.

Definition at line 83 of file content.c.

References content, inx_find(), M_TYPE_STRINGER, NULLER, and http_content_t::type.

Referenced by http_page_get(), register_print_message(), register_print_step1(), register_print_step2(), and register_print_step3().

5.423.1.9 bool_t http_load_file (int_t template, chr_t * filename)

Load file content into the http server repository.

Note:

Each file that is loaded will be cached for retrieval by http clients, with its mime type determined automatically. Any files passed as a template will have the ".template" extension trimmed from the end of its resource path, and any file named 'index.html' will be read as the default directory index path. Each file will also be added to its respective parent page or template inx holder.

Parameters:

template a value specifying whether or not the specified file is a template.

filename a null-terminated string specifying the pathname of the file to be loaded.

Returns:

0 on failure or 1 on success.

Definition at line 149 of file content.c.

References magma_t::content, content, CONTIGUOUS, data, HEAP, magma_t::http, http_free_content(), inx_insert(), magma_t::log, log_info, log_pedantic, M_TYPE_STRINGER, magma, MANAGED_T, MEMORYBUF, mm_alloc(), ns_length_get(), NULLER, PLACER, multi_t::st, st_alloc(), st_char_get(), st_cmp_ci_ends(), st_cmp_cs_ends(), st_cmp_cs_eq(), st_dupe(), st_dupe_opts(), st_free(), st_import(), st_length_get(), st_length_int(), st_length_set(), and multi_t::val.

Referenced by http_content_load_directory().

5.423.1.10 void http_page_free (http_page_t * page)

Free an http page object and all its underlying data.

Parameters:

page a pointer to the http page object to be freed.

Definition at line 43 of file content.c.

References `http_page_t::doc_ctx`, `http_page_t::doc_obj`, `mm_free()`, `xml_free_doc()`, `xml_free_parser_ctx()`, `xml_free_xpath_ctx()`, and `http_page_t::xpath_ctx`.

Referenced by `contact_print_form()`, `contact_print_message()`, `http_page_get()`, `portal_print_login()`, `statistics_process()`, `teacher_print_form()`, and `teacher_print_message()`.

5.423.1.11 http_page_t* http_page_get (chr_t * location)

Get a template page and prepare its xml document root for use.

Note:

Each page is affixed with an xpath with a namespace after passing through the xml parser.

Parameters:

location a pointer to a null-terminated string with the pathname of the template to be returned.

Returns:

NULL on failure, or a pointer to the http page object of the requested template.

Definition at line 100 of file content.c.

References `http_page_t::content`, `http_page_t::doc_ctx`, `http_page_t::doc_obj`, `http_get_template()`, `http_page_free()`, `log_pedantic`, `mm_alloc()`, `http_content_t::resource`, `st_char_get()`, `st_length_get()`, `xml_create_doc()`, `xml_create_parser_ctx()`, `xml_create_xpath_ctx()`, `xml_xpath_set_namespace()`, and `http_page_t::xpath_ctx`.

Referenced by `contact_business_add_error()`, `contact_print_form()`, `contact_print_message()`, `portal_print_login()`, `statistics_process()`, `teacher_add_error()`, `teacher_print_form()`, and `teacher_print_message()`.

5.423.2 Variable Documentation**5.423.2.1 struct { ... } content**

Referenced by `http_content_load_fonts()`, `http_content_refresh()`, `http_content_start()`, `http_content_stop()`, `http_get_static()`, `http_get_template()`, `http_load_file()`, `http_response()`, `mail_discover_encoding()`, `mail_discover_type()`, `mail_get_boundary()`, `mail_mime_boundary()`, `portal_endpoint_config_edit()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_edit()`, `register_print_message()`, `register_print_step1()`, `register_print_step2()`, and `register_print_step3()`.

5.423.2.2 inx_t* fonts

Definition at line 11 of file content.c.

5.423.2.3 inx_t * pages

Definition at line 11 of file content.c.

5.423.2.4 inx_t * templates

Definition at line 11 of file content.c.

5.424 src/servers/http/http.c File Reference

```
#include "magma.h"
```

Functions

- void [http_close](#) ([connection_t](#) *con)
Close a connection corresponding to an http session.
- void [http_requeue](#) ([connection_t](#) *con)
The main http server requeue entry point state machine for processing client data.
- void [http_body](#) ([connection_t](#) *con)
Get the body of the http request by reading the value of the Content-Length header.
- void [http_process](#) ([connection_t](#) *con)
Process data sent by an http client.
- void [http_init](#) ([connection_t](#) *con)
Handle a new http client connection.

5.424.1 Function Documentation

5.424.1.1 void [http_body](#) ([connection_t](#) * con)

Get the body of the http request by reading the value of the Content-Length header. [http.c](#)

Note:

Any request errors will be handled directly without returns. This function sets the value of the connection's `http.body` member.

Parameters:

con a pointer to the connection object of the remote http client.

Returns:

This function returns no value.

Definition at line 73 of file `http.c`.

References `con_read()`, `CONTIGUOUS`, `data`, `HEAP`, `magma_t::http`, `http_data_get()`, `HTTP_DATA_HEADER`, `HTTP_ERROR_400`, `HTTP_RESPOND`, `JOINTED`, `length`, `magma_t::log`, `log_pedantic`, `magma`, `MANAGED_T`, `MAPPED_T`, `size_conv_bl()`, `st_alloc_opts()`, `st_append_opts()`, `st_char_get()`, `st_cleanup`, `st_data_get()`, `st_dupe_opts()`, `st_empty`, `st_length_get()`, `st_length_int()`, and `http_data_t::value`.

Referenced by `http_requeue()`.

5.424.1.2 void [http_close](#) ([connection_t](#) * con)

Close a connection corresponding to an http session.

Returns:

This function returns no value.

Definition at line 14 of file http.c.

References `con_destroy()`.

Referenced by `http_process()`, and `http_requeue()`.

5.424.1.3 void http_init (connection_t * con)

Handle a new http client connection.

Parameters:

con a pointer to the http client connection that was just accepted.

Returns:

This function returns no value.

Definition at line 178 of file http.c.

References `con_reverse_enqueue()`, and `http_process()`.

Referenced by `protocol_enqueue()`.

5.424.1.4 void http_process (connection_t * con)

Process data sent by an http client.

Note:

This is performed in two stages: first read the http method with [http_parse_method\(\)](#), then transfer control to [http_parse_header\(\)](#). If a failure occurs reading a line of data, or too many protocol violations occur, [http_close\(\)](#) is called to drop the connection.

Parameters:

con the connection object from which to read data.

Returns:

This function returns no value.

Definition at line 137 of file http.c.

References `con_read_line()`, `enqueue()`, `http_close()`, `http_parse_header()`, `HTTP_PARSE_HEADER`, `http_parse_method()`, `http_process()`, `HTTP_READY`, `http_requeue()`, `log_pedantic`, `pl_empty()`, and `requeue()`.

Referenced by `http_init()`, `http_process()`, and `http_requeue()`.

5.424.1.5 void http_requeue (connection_t * con)

The main http server requeue entry point state machine for processing client data.

Returns:

This function returns no value.

Definition at line 24 of file http.c.

References `con_status()`, `enqueue()`, `http_body()`, `http_close()`, `HTTP_CLOSE`, `HTTP_COMPLETE`, `HTTP_ERROR_400`, `HTTP_ERROR_403`, `HTTP_ERROR_404`, `HTTP_ERROR_405`, `HTTP_ERROR_500`, `HTTP_ERROR_501`, `http_parse_pairs()`, `HTTP_PARSE_PAIRS`, `http_print_400()`, `http_print_403()`, `http_print_404()`, `http_print_405()`, `http_print_500()`, `http_print_501()`, `http_process()`, `HTTP_READ_BODY`, `http_requeue()`, `HTTP_RESPOND`, `http_response()`, `http_session_reset()`, `requeue()`, and `status`.

Referenced by `http_process()`, and `http_requeue()`.

5.425 src/servers/http/response.c File Reference

```
#include "magma.h"
```

Functions

- [chr_t * http_response_status](#) ([int_t](#) status)
Get a descriptive string for a numerical http status code.
- [stringer_t * http_response_cookie](#) ([connection_t](#) *con)
- [stringer_t * http_response_allow_cross](#) ([connection_t](#) *con)
- [stringer_t * http_response_connection](#) ([connection_t](#) *con, [int_t](#) force)
Get an appropriate value for the Connection header of an http response.
- void [http_response_options](#) ([connection_t](#) *con)
Return a response to an http OPTIONS request.
- void [http_response_header](#) ([connection_t](#) *con, [int_t](#) status, [stringer_t](#) *type, [size_t](#) len)
Send a full set of http response headers to the remote client.
- void [http_response](#) ([connection_t](#) *con)
Make a response to an http client request.

5.425.1 Function Documentation

5.425.1.1 void http_response (connection_t * con)

Make a response to an http client request. [response.c](#)

Note:

The following http methods aren't supported: PUT, DELETE, HEAD, TRACE, and CONNECT. The http server will first attempt to retrieve the requested url as a static page; otherwise the following special locations are supported: /portal, /portal/camel, /register, /contact, /report_abuse, /teacher, and /statistics.

Returns:

This function returns no value.

Definition at line 419 of file response.c.

References [magma_t::abuse](#), [magma_t::admin](#), [con_write_st\(\)](#), [magma_t::contact](#), [contact_process\(\)](#), [content](#), [HTTP_ERROR_403](#), [HTTP_ERROR_404](#), [HTTP_ERROR_405](#), [HTTP_ERROR_500](#), [HTTP_ERROR_501](#), [http_get_static\(\)](#), [HTTP_METHOD_CONNECT](#), [HTTP_METHOD_DELETE](#), [HTTP_METHOD_HEAD](#), [HTTP_METHOD_OPTIONS](#), [HTTP_METHOD_POST](#), [HTTP_METHOD_PUT](#), [HTTP_METHOD_TRACE](#), [HTTP_METHOD_UNSUPPORTED](#), [http_parse_pairs\(\)](#), [HTTP_READ_BODY](#), [HTTP_RESPOND](#), [http_response_header\(\)](#), [http_response_options\(\)](#), [json_api_dispatch\(\)](#), [magma](#), [NULLER](#), [PLACER](#), [portal_endpoint\(\)](#), [portal_process\(\)](#), [portal_upload\(\)](#), [register_process\(\)](#), [magma_t::registration](#), [http_content_t::resource](#), [st_cmp_ci_starts\(\)](#), [st_cmp_cs_eq\(\)](#), [st_cmp_cs_starts\(\)](#), [st_length_get\(\)](#), [magma_t::statistics](#), [statistics_process\(\)](#), [teacher_process\(\)](#), [http_content_t::type](#), and [magma_t::web](#).

Referenced by [http_queue\(\)](#).

5.425.1.2 stringer_t* http_response_allow_cross (connection_t * con)

Definition at line 245 of file response.c.

References `http_data_get()`, `HTTP_DATA_HEADER`, `http_parse_origin()`, `MANAGEDBUF`, `PLACER`, `placer_t`, `st_append`, `st_char_get()`, `st_cmp_ci_eq()`, `st_length_int()`, `st_quick()`, `upper_st()`, and `http_data_t::value`.

Referenced by `http_response_header()`, and `http_response_options()`.

5.425.1.3 stringer_t* http_response_connection (connection_t * con, int_t force)

Get an appropriate value for the Connection header of an http response.

Note:

If `magma.http.close` was set in the configuration options, the connection will be closed immediately.

Parameters:

con a pointer to the connection object of the outgoing response.

force either `HTTP_CONNECTION_CLOSE`, `HTTP_CONNECTION_NEUTRAL`, or `HTTP_CONNECTION_KEEPALIVE`. `HTTP_CONNECTION_CLOSE` will result in the connection being terminated immediately, `HTTP_CONNECTION_KEEPALIVE` will result in a "Connection: keep-alive" response, and `HTTP_CONNECTION_NEUTRAL` will not result in any output.

Returns:

a pointer to a managed string containing the http Connection header field that should be used for the response.

Definition at line 288 of file response.c.

References `CONTIGUOUS`, `HEAP`, `magma_t::http`, `HTTP_CLOSE`, `HTTP_CONNECTION_CLOSE`, `HTTP_CONNECTION_KEEPALIVE`, `magma`, `MANAGED_T`, `PLACER`, and `st_dupe_opts()`.

Referenced by `http_response_header()`, and `http_response_options()`.

5.425.1.4 stringer_t* http_response_cookie (connection_t * con)

Definition at line 190 of file response.c.

References `con_secure()`, `HEAP`, `HTTP_COOKIE_DELETE`, `HTTP_COOKIE_SET`, `JOINTED`, `MANAGED_T`, `MANAGEDBUF`, `PLACER`, `st_append`, `st_aprint_opts()`, `st_char_get()`, `st_length_int()`, `st_quick()`, and `time_print_gmt()`.

Referenced by `http_response_header()`.

5.425.1.5 void http_response_header (connection_t * con, int_t status, stringer_t * type, size_t len)

Send a full set of http response headers to the remote client.

Note:

If the mode is `HTTP_RESPOND` it will be changed to `HTTP_COMPLETE`, to tell the http requeue function to reset the context and enqueue request processor.

Parameters:

con a pointer to the connection object across which the response will be sent.

status an integer containing the http status code for the response.

type a managed string containing the value of the Content-Type header.

len the value of the Content-Length header.

Returns:

This function returns no value.

Definition at line 364 of file response.c.

References `con_print()`, `magma_t::http`, `HTTP_COMPLETE`, `HTTP_CONNECTION_NEUTRAL`, `HTTP_RESPOND`, `http_response_allow_cross()`, `http_response_connection()`, `http_response_cookie()`, `http_response_status()`, `magma`, `MANAGEDBUF`, `mm_wipe()`, `st_char_get()`, `st_cleanup`, `st_length_int()`, and `time_print_gmt()`.

Referenced by `api_response()`, `contact_print_form()`, `contact_print_message()`, `http_print_400()`, `http_print_403()`, `http_print_404()`, `http_print_405()`, `http_print_500()`, `http_print_500_log()`, `http_print_501()`, `http_response()`, `portal_endpoint()`, `portal_endpoint_attachments_progress()`, `portal_endpoint_error()`, `portal_endpoint_response()`, `portal_endpoint_search()`, `portal_print_login()`, `register_print_message()`, `register_print_step1()`, `register_print_step2()`, `register_print_step3()`, `statistics_process()`, and `teacher_print_form()`.

5.425.1.6 void http_response_options (connection_t * con)

Return a response to an http OPTIONS request.

Parameters:

con the client connection which made the OPTIONS request.

Returns:

This function returns no value.

LOW: I couldn't find a definitive answer about whether the OPTION response should always return a Content-Type of text/plain or use the Content-Type associated with the location provided in the request.

Definition at line 316 of file response.c.

References `build_stamp()`, `build_version()`, `con_print()`, `magma_t::http`, `HTTP_COMPLETE`, `HTTP_CONNECTION_NEUTRAL`, `HTTP_RESPOND`, `http_response_allow_cross()`, `http_response_connection()`, `magma`, `MANAGEDBUF`, `mm_wipe()`, `st_char_get()`, `st_cleanup`, `st_length_int()`, and `time_print_gmt()`.

Referenced by `http_response()`.

5.425.1.7 chr_t* http_response_status (int_t status)

Get a descriptive string for a numerical http status code.

Parameters:

status the value of the http status code to be looked up.

Returns:

NULL on failure, or a pointer to a null-terminated string containing a description of the specified http status code on success.

Definition at line 15 of file response.c.

References `log_pedantic`.

Referenced by `http_response_header()`.

5.426 src/servers/imap/fetch.c File Reference

```
#include "magma.h"
```

Functions

- [int_t imap_valid_sequence](#) ([stringer_t](#) *range)
- [void imap_fetch_free_items](#) ([imap_fetch_dataitems_t](#) *items)
- [imap_fetch_dataitems_t](#) * [imap_parse_dataitems](#) ([imap_arguments_t](#) *arguments)
- [stringer_t](#) * [imap_fetch_envelope](#) ([stringer_t](#) *header)
- [stringer_t](#) * [imap_fetch_bodystructure](#) ([mail_mime_t](#) *mime)
- [stringer_t](#) * [imap_fetch_body_header](#) ([placer_t](#) header, [imap_arguments_t](#) *array, [int_t](#) not)
- [stringer_t](#) * [imap_fetch_body_mime](#) ([placer_t](#) header)
- [stringer_t](#) * [imap_fetch_body_tag](#) ([stringer_t](#) *tag, [array_t](#) *items)
- [placer_t](#) [imap_fetch_body_portion](#) ([stringer_t](#) *part)
- [mail_mime_t](#) * [imap_fetch_body_part](#) ([mail_message_t](#) *message, [placer_t](#) portion)
- [int_t](#) [imap_fetch_parse_partial](#) ([stringer_t](#) *partial, [size_t](#) *start, [size_t](#) *length)
- [stringer_t](#) * [imap_fetch_return_header](#) ([connection_t](#) *con, [meta_message_t](#) *meta, [mail_message_t](#) **message, [stringer_t](#) **header, [imap_fetch_response_t](#) *output)
- [stringer_t](#) * [imap_fetch_return_text](#) ([connection_t](#) *con, [meta_message_t](#) *meta, [mail_message_t](#) **message, [stringer_t](#) **header, [imap_fetch_response_t](#) *output)
- [mail_message_t](#) * [imap_fetch_return_message](#) ([connection_t](#) *con, [meta_message_t](#) *meta, [mail_message_t](#) **message, [stringer_t](#) **header, [imap_fetch_response_t](#) *output)
- [mail_mime_t](#) * [imap_fetch_return_mime](#) ([connection_t](#) *con, [meta_message_t](#) *meta, [mail_message_t](#) **message, [stringer_t](#) **header, [imap_fetch_response_t](#) *output)
- [imap_fetch_response_t](#) * [imap_fetch_body](#) ([array_t](#) *outer, [array_t](#) *partial, [connection_t](#) *con, [meta_message_t](#) *meta, [mail_message_t](#) **message, [stringer_t](#) **header, [imap_fetch_response_t](#) *output)
- [imap_fetch_response_t](#) * [imap_fetch_message](#) ([connection_t](#) *con, [meta_message_t](#) *meta, [imap_fetch_dataitems_t](#) *items)
- [inx_t](#) * [imap_duplicate_messages](#) ([inx_t](#) *messages)
Create a copy of a collection of messages.
- [inx_t](#) * [imap_narrow_messages](#) ([inx_t](#) *messages, [uint64_t](#) selected, [stringer_t](#) *range, [int_t](#) uid)

5.426.1 Function Documentation

5.426.1.1 [inx_t](#)* [imap_duplicate_messages](#) ([inx_t](#) * *messages*)

Create a copy of a collection of messages. [fetch.c](#)

See also:

[meta_message_dupe\(\)](#)

Parameters:

messages a collection of meta message objects to be duplicated.

Returns:

the copied collection of meta messages on success, or NULL on failure.

Definition at line 1240 of file [fetch.c](#).

References [inx_alloc\(\)](#), [inx_append\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [inx_free\(\)](#), [inx_t](#), [log_error](#), [M_INX_LINKED](#), [M_TYPE_UINT64](#), [meta_message_dupe\(\)](#), [meta_message_free\(\)](#), [meta_message_t](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [imap_fetch\(\)](#), and [imap_store\(\)](#).

5.426.1.2 **imap_fetch_response_t*** **imap_fetch_body** (**array_t** * *outer*, **array_t** * *partial*, **connection_t** * *con*, **meta_message_t** * *meta*, **mail_message_t** ** *message*, **stringer_t** ** *header*, **imap_fetch_response_t** * *output*)

Definition at line 762 of file fetch.c.

References `ar_field_ar()`, `ar_field_type()`, `ar_length_get()`, `ARRAY_TYPE_ARRAY`, `mail_mime_t::body`, `mail_mime_t::header`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_fetch_body_header()`, `imap_fetch_body_mime()`, `imap_fetch_body_part()`, `imap_fetch_body_portion()`, `imap_fetch_body_tag()`, `imap_fetch_parse_partial()`, `imap_fetch_response_add()`, `imap_fetch_response_free()`, `imap_fetch_return_header()`, `imap_fetch_return_message()`, `imap_fetch_return_mime()`, `imap_fetch_return_text()`, `imap_get_ar_ar()`, `imap_get_ptr()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `length`, `mail_destroy()`, `mail_destroy_header()`, `mail_message_t::mime`, `number`, `pl_data_get()`, `pl_empty()`, `pl_init()`, `pl_length_get()`, `pl_null()`, `PLACER`, `placer_t`, `st_char_get()`, `st_cleanup`, `st_cmp_ci_eq()`, `st_free()`, `st_import()`, `st_length_get()`, and `st_merge`.

Referenced by `imap_fetch_message()`.

5.426.1.3 **stringer_t*** **imap_fetch_body_header** (**placer_t** *header*, **imap_arguments_t** * *array*, **int_t** *not*)

Definition at line 386 of file fetch.c.

References `ar_length_get()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_st_ar()`, `imap_get_type_ar()`, `mail_header_pop()`, `mm_cmp_ci_eq()`, `number`, `pl_char_get()`, `pl_data_get()`, `pl_empty()`, `placer_t`, `st_char_get()`, `st_cleanup`, `st_length_get()`, and `st_merge`.

Referenced by `imap_fetch_body()`.

5.426.1.4 **stringer_t*** **imap_fetch_body_mime** (**placer_t** *header*)

Definition at line 437 of file fetch.c.

References `mail_header_fetch_all()`, `PLACER`, `st_cleanup`, and `st_merge`.

Referenced by `imap_fetch_body()`.

5.426.1.5 **mail_mime_t*** **imap_fetch_body_part** (**mail_message_t** * *message*, **placer_t** *portion*)

Definition at line 555 of file fetch.c.

References `ar_field_ptr()`, `mail_mime_t::children`, `mail_message_t::mime`, `number`, `placer_t`, `tok_get_count_st()`, `tok_get_st()`, and `uint32_conv_st()`.

Referenced by `imap_fetch_body()`.

5.426.1.6 **placer_t** **imap_fetch_body_portion** (**stringer_t** * *part*)

Definition at line 512 of file fetch.c.

References `length`, `pl_init()`, `pl_null()`, `st_char_get()`, and `st_empty_out()`.

Referenced by `imap_fetch_body()`.

5.426.1.7 **stringer_t*** **imap_fetch_body_tag** (**stringer_t** * *tag*, **array_t** * *items*)

Definition at line 464 of file fetch.c.

References `ar_length_get()`, `imap_get_st_ar()`, `number`, `st_cleanup`, `st_free()`, `st_import()`, `st_merge`, and `upper_st()`.

Referenced by `imap_fetch_body()`.

5.426.1.8 stringer_t* imap_fetch_bodystructure (mail_mime_t * mime)

Definition at line 229 of file fetch.c.

References ar_field_ptr(), ar_field_st(), ar_free(), ar_length_get(), mail_mime_t::body, mail_mime_t::children, mail_mime_t::header, imap_build_array(), imap_build_array_isliteral(), imap_fetch_bodystructure(), items, length, mail_header_fetch_cleaned(), mail_mime_content_encoding(), mail_mime_content_id(), mail_mime_type_group(), mail_mime_type_parameters(), mail_mime_type_sub(), ns_length_get(), pl_init(), PLACER, st_char_get(), st_cleanup, st_cmp_cs_eq(), st_data_get(), st_free(), st_import(), st_length_get(), st_merge, and upper_st().

Referenced by imap_fetch_bodystructure(), and imap_fetch_message().

5.426.1.9 stringer_t* imap_fetch_envelope (stringer_t * header)

Definition at line 184 of file fetch.c.

References imap_build_array(), imap_parse_address(), mail_header_fetch_cleaned(), pl_init(), pl_null(), PLACER, placer_t, st_char_get(), st_cleanup, and st_length_get().

Referenced by imap_fetch_message().

5.426.1.10 void imap_fetch_free_items (imap_fetch_dataitems_t * items)

Definition at line 32 of file fetch.c.

References mm_cleanup, mm_free(), imap_fetch_dataitems_t::normal, imap_fetch_dataitems_t::normal_partial, imap_fetch_dataitems_t::peek, and imap_fetch_dataitems_t::peek_partial.

Referenced by imap_fetch(), and imap_parse_dataitems().

5.426.1.11 imap_fetch_response_t* imap_fetch_message (connection_t * con, meta_message_t * meta, imap_fetch_dataitems_t * items)

Definition at line 1043 of file fetch.c.

References imap_fetch_dataitems_t::body, mail_mime_t::body, imap_fetch_dataitems_t::bodystructure, CONTIGUOUS, imap_fetch_dataitems_t::envelope, imap_fetch_dataitems_t::flags, HEAP, imap_fetch_body(), imap_fetch_bodystructure(), imap_fetch_envelope(), imap_fetch_response_add(), imap_fetch_response_free(), imap_fetch_return_header(), imap_fetch_dataitems_t::internaldate, log_pedantic, mail_destroy(), mail_destroy_header(), mail_load_message(), mail_mime_update(), MAIL_STATUS_ANSWERED, MAIL_STATUS_DELETED, MAIL_STATUS_DRAFT, MAIL_STATUS_FLAGGED, MAIL_STATUS_RECENT, MAIL_STATUS_SEEN, MANAGED_T, mail_message_t::mime, imap_fetch_dataitems_t::normal, imap_fetch_dataitems_t::normal_partial, ns_length_get(), imap_fetch_dataitems_t::peek, imap_fetch_dataitems_t::peek_partial, PLACER, imap_fetch_dataitems_t::rfc822, imap_fetch_dataitems_t::rfc822_header, imap_fetch_dataitems_t::rfc822_size, imap_fetch_dataitems_t::rfc822_text, st_aprint_opts(), st_import(), st_length_get(), st_merge, mail_message_t::text, and imap_fetch_dataitems_t::uid.

Referenced by imap_fetch().

5.426.1.12 int_t imap_fetch_parse_partial (stringer_t * partial, size_t * start, size_t * length)

Definition at line 609 of file fetch.c.

References int32_conv_bl(), number, st_char_get(), and st_length_get().

Referenced by imap_fetch_body().

5.426.1.13 stringer_t* imap_fetch_return_header (connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)

Definition at line 685 of file fetch.c.

References `imap_fetch_response_free()`, `mail_destroy()`, `mail_load_header()`, `st_char_get()`, and `st_import()`.

Referenced by `imap_fetch_body()`, and `imap_fetch_message()`.

5.426.1.14 `mail_message_t* imap_fetch_return_message (connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)`

Definition at line 728 of file `fetch.c`.

References `imap_fetch_response_free()`, `mail_destroy()`, `mail_destroy_header()`, `mail_load_message()`, and `mail_mime_update()`.

Referenced by `imap_fetch_body()`.

5.426.1.15 `mail_mime_t* imap_fetch_return_mime (connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)`

Definition at line 742 of file `fetch.c`.

References `imap_fetch_response_free()`, `mail_destroy()`, `mail_destroy_header()`, `mail_load_message()`, and `mail_mime_update()`.

Referenced by `imap_fetch_body()`.

5.426.1.16 `stringer_t* imap_fetch_return_text (connection_t * con, meta_message_t * meta, mail_message_t ** message, stringer_t ** header, imap_fetch_response_t * output)`

Definition at line 714 of file `fetch.c`.

References `imap_fetch_response_free()`, `mail_destroy()`, `mail_destroy_header()`, and `mail_load_message()`.

Referenced by `imap_fetch_body()`.

5.426.1.17 `inx_t* imap_narrow_messages (inx_t * messages, uint64_t selected, stringer_t * range, int_t uid)`

Definition at line 1279 of file `fetch.c`.

References `inx_alloc()`, `inx_append()`, `inx_cleanup()`, `inx_count()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `log_error`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `meta_message_t`, `number`, `pl_char_get()`, `pl_empty()`, `pl_null()`, `placer_t`, `st_char_get()`, `st_length_int()`, `tok_get_count_st()`, `tok_get_st()`, `multi_t::u64`, `uint64_conv_st()`, and `multi_t::val`.

Referenced by `imap_copy()`, `imap_fetch()`, and `imap_store()`.

5.426.1.18 `imap_fetch_dataitems_t* imap_parse_dataitems (imap_arguments_t * arguments)`

Definition at line 44 of file `fetch.c`.

References `ar_append()`, `ar_length_get()`, `ARRAY_TYPE_ARRAY`, `ARRAY_TYPE_POINTER`, `imap_fetch_dataitems_t::body`, `imap_fetch_dataitems_t::bodystructure`, `imap_fetch_dataitems_t::envelope`, `imap_fetch_dataitems_t::flags`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_fetch_free_items()`, `imap_get_ar_ar()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_fetch_dataitems_t::internaldate`, `log_error`, `mm_alloc()`, `imap_fetch_dataitems_t::normal`, `imap_fetch_dataitems_t::normal_partial`, `number`, `imap_fetch_dataitems_t::peek`, `imap_fetch_dataitems_t::peek_partial`, `PLACER`, `imap_fetch_dataitems_t::rfc822`, `imap_fetch_dataitems_t::rfc822_header`, `imap_fetch_dataitems_t::rfc822_size`, `imap_fetch_dataitems_t::rfc822_text`, `st_cmp_ci_eq()`, `st_cmp_cs_starts()`, `type()`, and `imap_fetch_dataitems_t::uid`.

Referenced by `imap_fetch()`.

5.426.1.19 `int_t imap_valid_sequence (stringer_t * range)`

Definition at line 11 of file `fetch.c`.

References `length`, and `st_empty_out()`.

Referenced by `imap_copy()`, `imap_fetch()`, `imap_search_messages_inner()`, and `imap_store()`.

5.427 src/servers/imap/fetch_response.c File Reference

```
#include "magma.h"
```

Functions

- void [imap_fetch_response_free](#) ([imap_fetch_response_t](#) *response)
- [imap_fetch_response_t](#) * [imap_fetch_response_add](#) ([imap_fetch_response_t](#) *response, [stringer_t](#) *key, [stringer_t](#) *value)
[fetch_response.c](#)

5.427.1 Function Documentation

5.427.1.1 [imap_fetch_response_t](#)* [imap_fetch_response_add](#) ([imap_fetch_response_t](#) * *response*, [stringer_t](#) * *key*, [stringer_t](#) * *value*)

[fetch_response.c](#)

Definition at line 25 of file [fetch_response.c](#).

References [CONTIGUOUS](#), [HEAP](#), [imap_fetch_response_t::key](#), [log_error](#), [MANAGED_T](#), [mm_alloc\(\)](#), [mm_free\(\)](#), [imap_fetch_response_t::next](#), [st_dup_opts\(\)](#), [st_free\(\)](#), and [imap_fetch_response_t::value](#).

Referenced by [imap_fetch_body\(\)](#), and [imap_fetch_message\(\)](#).

5.427.1.2 void [imap_fetch_response_free](#) ([imap_fetch_response_t](#) * *response*)

Definition at line 10 of file [fetch_response.c](#).

References [imap_fetch_response_t::key](#), [mm_free\(\)](#), [imap_fetch_response_t::next](#), [st_cleanup](#), and [imap_fetch_response_t::value](#).

Referenced by [imap_fetch\(\)](#), [imap_fetch_body\(\)](#), [imap_fetch_message\(\)](#), [imap_fetch_return_header\(\)](#), [imap_fetch_return_message\(\)](#), [imap_fetch_return_mime\(\)](#), and [imap_fetch_return_text\(\)](#).

5.428 src/servers/imap/flags.c File Reference

```
#include "magma.h"
```

Functions

- uint32_t [imap_get_flag](#) (stringer_t *string)
 - int_t [imap_flag_action](#) (stringer_t *string)
- flags.c*
- uint32_t [imap_flag_parse](#) (void *ptr, int_t type)
 - void [imap_update_flags](#) (meta_user_t *user, inx_t *messages, uint64_t foldernum, int_t action, uint32_t flags)

5.428.1 Function Documentation

5.428.1.1 int_t [imap_flag_action](#) (stringer_t * *string*)

flags.c

Definition at line 41 of file flags.c.

References [IMAP_FLAG_ADD](#), [IMAP_FLAG_REMOVE](#), [IMAP_FLAG_REPLACE](#), [IMAP_FLAG_SILENT](#), [PLACER](#), and [st_cmp_ci_eq\(\)](#).

Referenced by [imap_store\(\)](#).

5.428.1.2 uint32_t [imap_flag_parse](#) (void * *ptr*, int_t *type*)

Definition at line 65 of file flags.c.

References [ar_length_get\(\)](#), [IMAP_ARGUMENT_TYPE_ARRAY](#), [imap_get_flag\(\)](#), [imap_get_st_ar\(\)](#), [imap_get_type_ar\(\)](#), [MAIL_STATUS_EMPTY](#), and [number](#).

Referenced by [imap_append\(\)](#), and [imap_store\(\)](#).

5.428.1.3 uint32_t [imap_get_flag](#) (stringer_t * *string*)

Definition at line 10 of file flags.c.

References [MAIL_STATUS_ANSWERED](#), [MAIL_STATUS_DELETED](#), [MAIL_STATUS_DRAFT](#), [MAIL_STATUS_FLAGGED](#), [MAIL_STATUS_RECENT](#), [MAIL_STATUS_SEEN](#), [PLACER](#), and [st_cmp_ci_eq\(\)](#).

Referenced by [imap_flag_parse\(\)](#).

5.428.1.4 void [imap_update_flags](#) (meta_user_t * *user*, inx_t * *messages*, uint64_t *foldernum*, int_t *action*, uint32_t *flags*)

Definition at line 112 of file flags.c.

References [IMAP_FLAG_ADD](#), [IMAP_FLAG_REMOVE](#), [IMAP_FLAG_REPLACE](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [log_error](#), [MAIL_STATUS_ANSWERED](#), [MAIL_STATUS_DELETED](#), [MAIL_STATUS_DRAFT](#), [MAIL_STATUS_EMPTY](#), [MAIL_STATUS_FLAGGED](#), [MAIL_STATUS_RECENT](#), [MAIL_STATUS_SEEN](#), [meta_data_flags_add\(\)](#), [meta_data_flags_remove\(\)](#), [meta_data_flags_replace\(\)](#), and [meta_message_t](#).

Referenced by [imap_store\(\)](#).

5.429 src/web/portal/flags.c File Reference

```
#include "magma.h"
```

Functions

- `json_t * portal_message_flags_array (meta_message_t *meta)`
Return a json array of flag descriptions for a mail message's flags bitmask.
- `int_t portal_parse_flags (json_t *array, uint32_t *flags)`
- `json_t * portal_message_tags_array (meta_message_t *meta)`

5.429.1 Function Documentation

5.429.1.1 `json_t* portal_message_flags_array (meta_message_t * meta)`

Return a json array of flag descriptions for a mail message's flags bitmask. flags.c

TODO: The messages.load method uses the flags/tags helper functions, but the messages.list and the messages.tags/flags methods still need to be updated.

Parameters:

meta a pointer to the meta message object of the mail message to have its flags parsed.

Returns:

NULL on failure, or a pointer to the json object of the specified mail message's flags.

Definition at line 18 of file flags.c.

References `json_array_append_new_d`, `json_array_d`, `json_string_d`, `MAIL_MARK_BLACKHOLED`, `MAIL_MARK_INFECTED`, `MAIL_MARK_JUNK`, `MAIL_MARK_PHISHING`, `MAIL_MARK_SPOOFED`, `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_APPENDED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_EMPTY`, `MAIL_STATUS_ENCRYPTED`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, and `MAIL_STATUS_SEEN`.

Referenced by `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, and `portal_message_meta()`.

5.429.1.2 `json_t* portal_message_tags_array (meta_message_t * meta)`

Definition at line 86 of file flags.c.

References `ar_field_st()`, `ar_length_get()`, `count`, `json_array_append_new_d`, `json_array_d`, `json_string_d`, and `st_char_get()`.

Referenced by `portal_message_meta()`.

5.429.1.3 `int_t portal_parse_flags (json_t * array, uint32_t * flags)`

Definition at line 46 of file flags.c.

References `count`, `json_array_get_d`, `json_array_size_d`, `json_string_value_d`, `MAIL_MARK_BLACKHOLED`, `MAIL_MARK_INFECTED`, `MAIL_MARK_JUNK`, `MAIL_MARK_PHISHING`, `MAIL_MARK_SPOOFED`, `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_APPENDED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `NULLER`, `PLACER`, and `st_cmp_ci_eq()`.

Referenced by `portal_endpoint_messages_flag()`.

5.430 src/servers/imap/imap.c File Reference

```
#include "magma.h"
```

Functions

- void [imap_starttls](#) ([connection_t](#) *con)

Create a secure connection for an IMAP session.

- void [imap_invalid](#) ([connection_t](#) *con)

Respond to an invalid IMAP command from a client.

- void [imap_logout](#) ([connection_t](#) *con)

Terminate an IMAP session gracefully with a "BYE" message and destroy the underlying connection.

- void [imap_login](#) ([connection_t](#) *con)

Attempt to perform a user login on an IMAP client connection.

- void [imap_noop](#) ([connection_t](#) *con)

- void [imap_check](#) ([connection_t](#) *con)

- void [imap_list](#) ([connection_t](#) *con)

- void [imap_lsub](#) ([connection_t](#) *con)

- void [imap_create](#) ([connection_t](#) *con)

Create a new IMAP folder in response to the IMAP "CREATE" command.

- void [imap_delete](#) ([connection_t](#) *con)

Handle the IMAP "DELETE" command and delete the specified IMAP folder.

- void [imap_rename](#) ([connection_t](#) *con)

- void [imap_status](#) ([connection_t](#) *con)

- void [imap_subscribe](#) ([connection_t](#) *con)

- void [imap_unsubscribe](#) ([connection_t](#) *con)

- void [imap_examine](#) ([connection_t](#) *con)

- void [imap_select](#) ([connection_t](#) *con)

- void [imap_store](#) ([connection_t](#) *con)

- void [imap_close](#) ([connection_t](#) *con)

- void [imap_expunge](#) ([connection_t](#) *con)

- void [imap_copy](#) ([connection_t](#) *con)

- void [imap_append](#) ([connection_t](#) *con)

imap.c

- void [imap_fetch](#) ([connection_t](#) *con)

- void [imap_search](#) ([connection_t](#) *con)

- void [imap_idle](#) ([connection_t](#) *con)

- void [imap_id](#) ([connection_t](#) *con)

- void [imap_capability](#) ([connection_t](#) *con)

Display the capability string for the IMAP server.

- void [imap_init](#) ([connection_t](#) *con)

The main IMAP entry point for all inbound client connections, as dispatched by the generic protocol handler (display banner greeting).

5.430.1 Function Documentation

5.430.1.1 void imap_append (connection_t * con)

[imap.c](#)

BUG: If the user is over their quota check whether rollout is enabled and if so. If enabled make room for the appended message using the rollout logic.

Definition at line 1462 of file [imap.c](#).

References [ar_length_get\(\)](#), [con_print\(\)](#), [imap_append_message\(\)](#), [IMAP_ARGUMENT_TYPE_ARRAY](#), [IMAP_ARGUMENT_TYPE_LITERAL](#), [imap_flag_parse\(\)](#), [imap_get_ptr\(\)](#), [imap_get_st_ar\(\)](#), [imap_get_type_ar\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [log_pedantic](#), [MAIL_STATUS_EMPTY](#), [MAIL_STATUS_RECENT](#), [meta_folder_t](#), [meta_folders_by_name\(\)](#), [meta_message_t](#), [META_USER_OVERQUOTA](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

5.430.1.2 void imap_capability (connection_t * con)

Display the capability string for the IMAP server.

Note:

This string always needs to stay in sync with the banner greeting.

Returns:

This function returns no value.

Definition at line 1831 of file [imap.c](#).

References [con_print\(\)](#), [con_secure\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

5.430.1.3 void imap_check (connection_t * con)

Definition at line 297 of file [imap.c](#).

References [con_print\(\)](#), [imap_session_update\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

5.430.1.4 void imap_close (connection_t * con)

Definition at line 1087 of file [imap.c](#).

References [ar_length_get\(\)](#), [con_print\(\)](#), [imap_message_expunge\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [MAIL_STATUS_DELETED](#), [MAIL_STATUS_RECENT](#), [meta_message_t](#), [meta_messages_update_sequences\(\)](#), [meta_user_rlock\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_MESSAGES](#), [serial_get\(\)](#), [serial_increment\(\)](#), [st_char_get\(\)](#), [st_length_int\(\)](#), [user_lock\(\)](#), and [user_unlock\(\)](#).

5.430.1.5 void imap_copy (connection_t * con)

Definition at line 1296 of file [imap.c](#).

References [ar_length_get\(\)](#), [con_print\(\)](#), [count](#), [IMAP_ARGUMENT_TYPE_ARRAY](#), [imap_get_st_ar\(\)](#), [imap_get_type_ar\(\)](#), [imap_message_copier\(\)](#), [imap_narrow_messages\(\)](#), [imap_range_build\(\)](#), [imap_valid_sequence\(\)](#), [inx_count\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [inx_free\(\)](#), [inx_t](#), [log_pedantic](#), [MAIL_STATUS_RECENT](#), [meta_folder_t](#), [meta_folders_by_name\(\)](#), [meta_message_t](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [mm_alloc\(\)](#), [mm_free\(\)](#), [st_char_get\(\)](#), [st_length_get\(\)](#), [st_length_int\(\)](#), [user_lock\(\)](#), and [user_unlock\(\)](#).

5.430.1.6 void imap_create (connection_t * con)

Create a new IMAP folder in response to the IMAP "CREATE" command.

See also:

[imap_folder_create\(\)](#)

Parameters:

con the connection across which the folder creation request was made.

Returns:

This function returns no value.

Definition at line 471 of file imap.c.

References [ar_length_get\(\)](#), [con_print\(\)](#), [FOLDER_LENGTH_LIMIT](#), [IMAP_ARGUMENT_TYPE_ARRAY](#), [imap_folder_create\(\)](#), [IMAP_FOLDER_RECURSION_LMIIT](#), [imap_get_st_ar\(\)](#), [imap_get_type_ar\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_FOLDERS](#), [serial_get\(\)](#), [serial_increment\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

5.430.1.7 void imap_delete (connection_t * con)

Handle the IMAP "DELETE" command and delete the specified IMAP folder.

See also:

[imap_folder_remove\(\)](#)

Parameters:

con a pointer to the connection object of the IMAP session generating the delete request.

Returns:

This function returns no value.

Definition at line 538 of file imap.c.

References [ar_length_get\(\)](#), [con_print\(\)](#), [IMAP_ARGUMENT_TYPE_ARRAY](#), [imap_folder_remove\(\)](#), [imap_get_st_ar\(\)](#), [imap_get_type_ar\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_FOLDERS](#), [serial_get\(\)](#), [serial_increment\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

5.430.1.8 void imap_examine (connection_t * con)

Definition at line 802 of file imap.c.

References [ar_length_get\(\)](#), [con_print\(\)](#), [imap_folder_status_t::first](#), [imap_folder_status_t::foldernum](#), [IMAP_ARGUMENT_TYPE_ARRAY](#), [imap_folder_status\(\)](#), [imap_get_st_ar\(\)](#), [imap_get_type_ar\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [MAIL_STATUS_RECENT](#), [imap_folder_status_t::messages](#), [meta_message_t](#), [meta_user_rlock\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [imap_folder_status_t::recent](#), [st_char_get\(\)](#), [st_length_int\(\)](#), [status](#), and [imap_folder_status_t::uidnext](#).

5.430.1.9 void imap_expunge (connection_t * con)

Definition at line 1197 of file imap.c.

References [ar_length_get\(\)](#), [con_print\(\)](#), [imap_message_expunge\(\)](#), [imap_session_update\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [MAIL_STATUS_DELETED](#), [meta_message_t](#), [meta_messages_update_sequences\(\)](#), [meta_user_rlock\(\)](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [OBJECT_MESSAGES](#), [serial_get\(\)](#), [serial_increment\(\)](#), [st_char_get\(\)](#), [st_length_int\(\)](#), [user_lock\(\)](#), and [user_unlock\(\)](#).

5.430.1.10 void imap_fetch (connection_t * con)

Definition at line 1547 of file imap.c.

References `ar_length_get()`, `con_print()`, `con_status()`, `con_write_bl()`, `con_write_st()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_duplicate_messages()`, `imap_fetch_free_items()`, `imap_fetch_message()`, `imap_fetch_response_free()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_narrow_messages()`, `imap_parse_dataitems()`, `imap_valid_sequence()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_reset()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `items`, `imap_fetch_response_t::key`, `MAIL_STATUS_SEEN`, `meta_data_flags_add()`, `meta_message_t`, `meta_user_rlock()`, `meta_user_unlock()`, `meta_user_wlock()`, `imap_fetch_response_t::next`, `imap_fetch_dataitems_t::normal`, `OBJECT_MESSAGES`, `PLACER`, `imap_fetch_dataitems_t::rfc822`, `imap_fetch_dataitems_t::rfc822_header`, `imap_fetch_dataitems_t::rfc822_text`, `serial_get()`, `serial_increment()`, `st_char_get()`, `st_cmp_cs_eq()`, `st_length_int()`, `status`, and `imap_fetch_response_t::value`.

5.430.1.11 void imap_id (connection_t * con)

Definition at line 1782 of file imap.c.

References `ar_length_get()`, `build_stamp()`, `build_version()`, `con_addr_presentation()`, `con_print()`, `count`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_ar_ar()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `log_info`, `MANAGEDBUF`, `st_append_opts()`, `st_char_get()`, `st_free()`, `st_length_get()`, `st_length_int()`, `st_sprint()`, and `time_print_local()`.

5.430.1.12 void imap_idle (connection_t * con)

Definition at line 1765 of file imap.c.

References `con_print()`, `st_char_get()`, and `st_length_int()`.

5.430.1.13 void imap_init (connection_t * con)

The main IMAP entry point for all inbound client connections, as dispatched by the generic protocol handler (display banner greeting).

Parameters:

con a pointer to the connection object of the newly connected client.

Returns:

This function returns no value.

Definition at line 1853 of file imap.c.

References `build_version()`, `con_print()`, `con_reverse_enqueue()`, `con_secure()`, `imap_requeue()`, `st_char_get()`, `st_length_get()`, and `st_length_int()`.

Referenced by `protocol_enqueue()`.

5.430.1.14 void imap_invalid (connection_t * con)

Respond to an invalid IMAP command from a client.

Parameters:

con a pointer to the client connection that issued the bad command.

Returns:

This function returns no value.

Definition at line 63 of file imap.c.

References `con_print()`, `st_char_get()`, and `st_length_int()`.

Referenced by `imap_process()`, and `imap_starttls()`.

5.430.1.15 void imap_list (connection_t * con)

Definition at line 312 of file imap.c.

References `ar_length_get()`, `con_print()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_name_escaped()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_narrow_folders()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_reset()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `log_pedantic`, `magma_folder_name()`, `meta_folder_t`, `meta_user_rlock()`, `meta_user_unlock()`, `NULLER`, `PLACER`, `st_char_get()`, `st_cmp_ci_eq()`, `st_free()`, and `st_length_int()`.

5.430.1.16 void imap_login (connection_t * con)

Attempt to perform a user login on an IMAP client connection.

Parameters:

con a pointer to the connection object of the remote session.

Returns:

This function returns no value.

Definition at line 106 of file imap.c.

References `ar_length_get()`, `auth_free()`, `AUTH_LOCK_ABUSE`, `AUTH_LOCK_ADMIN`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_USER`, `auth_login()`, `cache_decrement()`, `cache_increment()`, `con_addr_presentation()`, `con_addr_subnet()`, `con_print()`, `con_secure()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_get_st_ar()`, `imap_get_type_ar()`, `inx_count()`, `auth_t::keys`, `lock_get()`, `lock_release()`, `auth_t::locked`, `log_info`, `MANAGEDBUF`, `auth_t::master`, `meta_data_update_log()`, `meta_get()`, `META_GET_FOLDERS`, `META_GET_KEYS`, `META_GET_MESSAGES`, `meta_inx_remove()`, `META_PROTOCOL_IMAP`, `META_USER_ENCRYPT_DATA`, `meta_user_rlock()`, `META_USER_TLS`, `meta_user_unlock()`, `auth_t::salt`, `auth_t::seasoning`, `st_char_get()`, `st_duped()`, `st_free()`, `st_length_int()`, `st_populated`, `st_quick()`, `auth_t::status`, `time_datestamp()`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, and `auth_t::verification`.

5.430.1.17 void imap_logout (connection_t * con)

Terminate an IMAP session gracefully with a "BYE" message and destroy the underlying connection.

Parameters:

con the IMAP connection to be terminated.

Returns:

This function returns no value.

Definition at line 79 of file imap.c.

References `con_destroy()`, `con_print()`, `con_status()`, `con_write_bl()`, `st_char_get()`, `st_length_int()`, and `status`.

Referenced by `imap_process()`, and `imap_requeue()`.

5.430.1.18 void imap_lsub (connection_t * con)

Definition at line 397 of file imap.c.

References `ar_length_get()`, `con_print()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_name_escaped()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_narrow_folders()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_reset()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `meta_folder_t`, `meta_user_rlock()`, `meta_user_unlock()`, `NULLER`, `PLACER`, `st_char_get()`, `st_cmp_ci_eq()`, `st_free()`, and `st_length_int()`.

5.430.1.19 `void imap_noop (connection_t * con)`

Definition at line 286 of file `imap.c`.

References `con_print()`, `imap_session_update()`, `st_char_get()`, and `st_length_int()`.

5.430.1.20 `void imap_rename (connection_t * con)`

Definition at line 590 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `FOLDER_LENGTH_LIMIT`, `IMAP_ARGUMENT_TYPE_ARRAY`, `IMAP_FOLDER_RECURSION_LMIIT`, `imap_folder_rename()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_FOLDERS`, `serial_get()`, `serial_increment()`, `st_char_get()`, and `st_length_int()`.

5.430.1.21 `void imap_search (connection_t * con)`

Definition at line 1712 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `con_write_st()`, `HEAP`, `imap_search_messages()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `JOINTED`, `MANAGED_T`, `MANAGEDBUF`, `meta_message_t`, `PLACER`, `st_append_opts()`, `st_aprint_opts()`, `st_char_get()`, `st_cleanup`, `st_length_int()`, `st_populated`, `st_sprint()`, and `status`.

5.430.1.22 `void imap_select (connection_t * con)`

Definition at line 873 of file `imap.c`.

References `ar_length_get()`, `con_print()`, `imap_folder_status_t::first`, `imap_folder_status_t::foldernum`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_status()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `MAIL_STATUS_RECENT`, `imap_folder_status_t::messages`, `meta_data_flags_remove()`, `meta_message_t`, `meta_user_unlock()`, `meta_user_wlock()`, `imap_folder_status_t::recent`, `st_char_get()`, `st_length_int()`, `status`, and `imap_folder_status_t::uidnext`.

5.430.1.23 `void imap_starttls (connection_t * con)`

Create a secure connection for an IMAP session. TODO: Review error messages and update them with the appropriate response code. LOW: When should we check the serial number to see if the local data is stale and needs to be refreshed? LOW: Add function descriptions to all of the different IMAP commands.

Note:

RFC 2595 / section 3.1 specifies that STARTTLS is only available in a non-authenticated state.

Parameters:

con the connection on top of which the TLS session will be established.

Returns:

This function returns no value.

Definition at line 23 of file `imap.c`.

References `con_print()`, `con_secure()`, `imap_invalid()`, `log_pedantic`, `M_SSL_BIO_NOCLOSE`, `pl_null()`, `st_char_get()`, `st_length_get()`, `st_length_set()`, `stats_increment_by_name()`, and `tls_server_alloc()`.

5.430.1.24 void imap_status (connection_t * con)

Definition at line 656 of file imap.c.

References `ar_length_get()`, `con_print()`, `imap_folder_status_t::foldernum`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_status()`, `imap_get_ar_ar()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_folder_status_t::messages`, `meta_user_rlock()`, `meta_user_unlock()`, `NULLER`, `number`, `PLACER`, `imap_folder_status_t::recent`, `st_append_opts()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_free()`, `st_length_get()`, `st_length_int()`, `status`, `imap_folder_status_t::uidnext`, `imap_folder_status_t::unseen`, and `values`.

5.430.1.25 void imap_store (connection_t * con)

LOW: Shouldn't we be checking for stale status info so the update doesn't make decisions based on incorrect status data? On the other hand the actual IMAP logic is passed all the way through to the DB so even if the server ends up with incorrect status information, the database should remain accurate.

Definition at line 952 of file imap.c.

References `ar_length_get()`, `con_print()`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_duplicate_messages()`, `imap_flag_action()`, `imap_flag_parse()`, `IMAP_FLAG_SILENT`, `imap_get_ptr()`, `imap_get_st_ar()`, `imap_get_type_ar()`, `imap_narrow_messages()`, `imap_session_update()`, `imap_update_flags()`, `imap_valid_sequence()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `meta_message_t`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_MESSAGES`, `serial_get()`, `serial_increment()`, `st_char_get()`, and `st_length_int()`.

5.430.1.26 void imap_subscribe (connection_t * con)

Definition at line 743 of file imap.c.

References `ar_length_get()`, `con_print()`, `FOLDER_LENGTH_LIMIT`, `IMAP_ARGUMENT_TYPE_ARRAY`, `imap_folder_create()`, `IMAP_FOLDER_RECURSION_LMIIT`, `imap_get_st_ar()`, `imap_get_type_ar()`, `meta_folder_t`, `meta_folders_by_name()`, `meta_user_unlock()`, `meta_user_wlock()`, `st_char_get()`, and `st_length_int()`.

5.430.1.27 void imap_unsubscribe (connection_t * con)

Definition at line 792 of file imap.c.

References `con_print()`, `st_char_get()`, and `st_length_int()`.

5.431 src/servers/imap/output.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * imap_build_array_isliteral \(placer_t data\)](#)
- [stringer_t * imap_build_array \(chr_t *format,...\)](#)
output.c

5.431.1 Function Documentation

5.431.1.1 stringer_t* imap_build_array (chr_t **format*, ...)

[output.c](#)

Definition at line 37 of file output.c.

References [imap_build_array_isliteral\(\)](#), [length](#), [log_error](#), [log_pedantic](#), [ns_length_get\(\)](#), [number](#), [pl_data_get\(\)](#), [pl_empty\(\)](#), [pl_init\(\)](#), [pl_length_get\(\)](#), [placer_t](#), [st_alloc\(\)](#), [st_char_get\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), and [st_length_set\(\)](#).

Referenced by [imap_fetch_bodystructure\(\)](#), [imap_fetch_envelope\(\)](#), [imap_parse_address\(\)](#), and [imap_parse_address_part\(\)](#).

5.431.1.2 stringer_t* imap_build_array_isliteral (placer_t *data*)

Definition at line 10 of file output.c.

References [length](#), [pl_data_get\(\)](#), [pl_empty\(\)](#), [pl_length_get\(\)](#), and [st_merge](#).

Referenced by [imap_build_array\(\)](#), and [imap_fetch_bodystructure\(\)](#).

5.432 src/servers/imap/parse_address.c File Reference

```
#include "magma.h"
```

Functions

- void [imap_parse_address_put](#) ([stringer_t](#) *buffer, [chr_t](#) c)
LOW: Do we need an [imap_parse_address_group\(\)](#) function?
- [stringer_t](#) * [imap_parse_address_part](#) ([placer_t](#) input)
- [placer_t](#) [imap_parse_address_breaker](#) ([stringer_t](#) *address, [uint32_t](#) part)
- [stringer_t](#) * [imap_parse_address](#) ([stringer_t](#) *address)
[parse_address.c](#)

5.432.1 Function Documentation

5.432.1.1 [stringer_t](#)* [imap_parse_address](#) ([stringer_t](#) * *address*)

[parse_address.c](#)

Definition at line 211 of file [parse_address.c](#).

References [imap_build_array\(\)](#), [imap_parse_address_breaker\(\)](#), [imap_parse_address_part\(\)](#), [pl_empty\(\)](#), [placer_t](#), [st_free\(\)](#), [st_merge](#), and [imap_fetch_response_t::value](#).

Referenced by [imap_fetch_envelope\(\)](#).

5.432.1.2 [placer_t](#) [imap_parse_address_breaker](#) ([stringer_t](#) * *address*, [uint32_t](#) *part*)

Definition at line 158 of file [parse_address.c](#).

References [length](#), [pl_init\(\)](#), [pl_null\(\)](#), [placer_t](#), [st_char_get\(\)](#), [st_length_get\(\)](#), and [status](#).

Referenced by [imap_parse_address\(\)](#).

5.432.1.3 [stringer_t](#)* [imap_parse_address_part](#) ([placer_t](#) *input*)

Definition at line 27 of file [parse_address.c](#).

References [imap_build_array\(\)](#), [imap_parse_address_put\(\)](#), [length](#), [pl_data_get\(\)](#), [pl_length_get\(\)](#), [placer_t](#), [st_alloc\(\)](#), [st_char_get\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [st_length_set\(\)](#), and [tok_get_st\(\)](#).

Referenced by [imap_parse_address\(\)](#).

5.432.1.4 void [imap_parse_address_put](#) ([stringer_t](#) * *buffer*, [chr_t](#) *c*)

LOW: Do we need an [imap_parse_address_group\(\)](#) function?

Definition at line 12 of file [parse_address.c](#).

References [st_avail_get\(\)](#), [st_char_get\(\)](#), [st_length_get\(\)](#), and [st_length_set\(\)](#).

Referenced by [imap_parse_address_part\(\)](#).

5.433 src/servers/imap/range.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * imap_range_build](#) (size_t *length*, uint64_t **numbers*)
[range.c](#)

5.433.1 Function Documentation

5.433.1.1 stringer_t* imap_range_build (size_t *length*, uint64_t **numbers*)

[range.c](#)

Definition at line 10 of file [range.c](#).

References [count](#), [log_pedantic](#), [st_free\(\)](#), and [st_merge](#).

Referenced by [imap_copy\(\)](#).

5.434 src/servers/molten/molten.c File Reference

```
#include "magma.h"
```

Functions

- void [molten_stats](#) ([connection_t](#) *con)
- void [molten_invalid](#) ([connection_t](#) *con)
- void [molten_quit](#) ([connection_t](#) *con)
- void [molten_init](#) ([connection_t](#) *con)

[molten.c](#)

5.434.1 Function Documentation

5.434.1.1 void [molten_init](#) ([connection_t](#) * *con*)

[molten.c](#)

Definition at line 53 of file [molten.c](#).

References [enqueue\(\)](#), and [molten_parse\(\)](#).

Referenced by [protocol_enqueue\(\)](#).

5.434.1.2 void [molten_invalid](#) ([connection_t](#) * *con*)

Definition at line 41 of file [molten.c](#).

References [con_write_bl\(\)](#), [enqueue\(\)](#), [molten_parse\(\)](#), and [molten_quit\(\)](#).

Referenced by [molten_parse\(\)](#).

5.434.1.3 void [molten_quit](#) ([connection_t](#) * *con*)

Definition at line 47 of file [molten.c](#).

References [con_destroy\(\)](#).

Referenced by [molten_invalid\(\)](#), [molten_parse\(\)](#), and [molten_stats\(\)](#).

5.434.1.4 void [molten_stats](#) ([connection_t](#) * *con*)

Definition at line 10 of file [molten.c](#).

References [con_print\(\)](#), [con_write_bl\(\)](#), [enqueue\(\)](#), [length](#), [molten_parse\(\)](#), [molten_quit\(\)](#), [stats_derived_count\(\)](#), [stats_derived_name\(\)](#), [stats_derived_value\(\)](#), [stats_get_count\(\)](#), [stats_get_name\(\)](#), and [stats_get_value_by_num\(\)](#).

5.435 src/servers/molten/molten.h File Reference

Functions

- void [molten_init](#) ([connection_t](#) *con)
molten.c
- void [molten_invalid](#) ([connection_t](#) *con)
- void [molten_quit](#) ([connection_t](#) *con)
- void [molten_stats](#) ([connection_t](#) *con)
- [int_t](#) [molten_compare](#) (const void *compare, const void *[command](#))
commands.c
- void [molten_parse](#) ([connection_t](#) *con)
- void [molten_sort](#) (void)
Sort the Molten command table to be ready for binary searches.
- void [molten_session_destroy](#) ([connection_t](#) *con)
sessions.c

5.435.1 Function Documentation

5.435.1.1 [int_t](#) [molten_compare](#) (const void * *compare*, const void * *command*)

commands.c

Definition at line 12 of file *commands.c*.

References [command_t](#), [PLACER](#), [st_cmp_ci_eq\(\)](#), and [st_cmp_ci_starts\(\)](#).

Referenced by [molten_parse\(\)](#), and [molten_sort\(\)](#).

5.435.1.2 void [molten_init](#) ([connection_t](#) * *con*)

molten.c

Definition at line 53 of file *molten.c*.

References [enqueue\(\)](#), and [molten_parse\(\)](#).

Referenced by [protocol_enqueue\(\)](#).

5.435.1.3 void [molten_invalid](#) ([connection_t](#) * *con*)

Definition at line 41 of file *molten.c*.

References [con_write_bl\(\)](#), [enqueue\(\)](#), [molten_parse\(\)](#), and [molten_quit\(\)](#).

Referenced by [molten_parse\(\)](#).

5.435.1.4 void [molten_parse](#) ([connection_t](#) * *con*)

Definition at line 32 of file *commands.c*.

References [command](#), [command_t](#), [con_read_line\(\)](#), [enqueue\(\)](#), [molten_commands](#), [molten_compare\(\)](#), [molten_invalid\(\)](#), [molten_parse\(\)](#), [molten_quit\(\)](#), [pl_char_get\(\)](#), [pl_empty\(\)](#), and [pl_length_get\(\)](#).

Referenced by [molten_init\(\)](#), [molten_invalid\(\)](#), [molten_parse\(\)](#), and [molten_stats\(\)](#).

5.435.1.5 void molten_quit (connection_t * con)

Definition at line 47 of file molten.c.

References `con_destroy()`.

Referenced by `molten_invalid()`, `molten_parse()`, and `molten_stats()`.

5.435.1.6 void molten_session_destroy (connection_t * con)

`sessions.c`

Definition at line 10 of file `sessions.c`.

Referenced by `con_destroy()`.

5.435.1.7 void molten_sort (void)

Sort the Molten command table to be ready for binary searches.

Returns:

This function returns no value.

Definition at line 27 of file `commands.c`.

References `command_t`, `molten_commands`, and `molten_compare()`.

Referenced by `protocol_init()`.

5.435.1.8 void molten_stats (connection_t * con)

Definition at line 10 of file molten.c.

References `con_print()`, `con_write_bl()`, `enqueue()`, `length`, `molten_parse()`, `molten_quit()`, `stats_derived_count()`, `stats_derived_name()`, `stats_derived_value()`, `stats_get_count()`, `stats_get_name()`, and `stats_get_value_by_num()`.

5.436 src/servers/pop/mailbox.c File Reference

```
#include "magma.h"
```

Functions

- `uint64_t pop_total_messages (inx_t *messages)`
Get the number of messages available to a POP3 user.
- `uint64_t pop_total_size (inx_t *messages)`
Get the total size of all messages available to a POP3 user.
- `uint64_t pop_get_last (inx_t *messages)`
Get the POP3 sequence number of the last message that isn't flagged as recent.
- `meta_message_t * pop_get_message (inx_t *messages, uint64_t get)`
Get a message by its pop sequence number.

5.436.1 Function Documentation

5.436.1.1 `uint64_t pop_get_last (inx_t * messages)`

Get the POP3 sequence number of the last message that isn't flagged as recent. [mailbox.c](#)

Note:

This function only counts messages that aren't deleted or hidden, and weren't created by the IMAP APPEND command.

Parameters:

messages an inx holder containing a collection of messages to be analyzed.

Returns:

the sequence number of the last message that isn't flagged as recent, or 0 on failure.

Definition at line 74 of file mailbox.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `MAIL_STATUS_APPENDED`, `MAIL_STATUS_HIDDEN`, `MAIL_STATUS_RECENT`, and `meta_message_t`.

Referenced by `pop_last()`.

5.436.1.2 `meta_message_t* pop_get_message (inx_t * messages, uint64_t get)`

Get a message by its pop sequence number.

Parameters:

messages an inx holder containing the collection of the user's messages to be traversed.
number the zero-based pop sequence number of the message to be retrieved.

Returns:

NULL on failure or the meta message object of the message if it was found.

Definition at line 113 of file mailbox.c.

References count, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_APPENDED, and meta_message_t.

Referenced by pop_delete(), pop_list(), pop_retr(), pop_top(), and pop_uidl().

5.436.1.3 uint64_t pop_total_messages (inx_t * messages)

Get the number of messages available to a POP3 user.

Note:

This function only counts messages that aren't deleted or hidden, and weren't created by the IMAP APPEND command.

Parameters:

messages an inx holder containing a collection of messages to be analyzed.

Returns:

the total number of messages available to the POP3 user, or 0 on failure.

Definition at line 16 of file mailbox.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_APPENDED, MAIL_STATUS_HIDDEN, and meta_message_t.

Referenced by pop_list(), pop_stat(), and pop_uidl().

5.436.1.4 uint64_t pop_total_size (inx_t * messages)

Get the total size of all messages available to a POP3 user.

Note:

This function only counts messages that aren't deleted or hidden, and weren't created by the IMAP APPEND command.

Parameters:

messages an inx holder containing a collection of messages to be analyzed.

Returns:

the total size, in bytes, of all messages available to the POP3 user, or 0 on failure.

Definition at line 45 of file mailbox.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_APPENDED, MAIL_STATUS_HIDDEN, and meta_message_t.

Referenced by pop_stat().

5.437 src/servers/pop/pop.c File Reference

```
#include "magma.h"
```

Functions

- void [pop_starttls](#) ([connection_t](#) *con)
TODO: Review error messages and update them with the appropriate response code, as per RFC 3206 regarding the response codes extension.
- void [pop_noop](#) ([connection_t](#) *con)
Execute a POP3 no-operation command.
- void [pop_invalid](#) ([connection_t](#) *con)
A function handler for invalid POP3 commands.
- void [pop_rset](#) ([connection_t](#) *con)
Reset the user's mailbox, in response to a POP3 RSET command.
- void [pop_quit](#) ([connection_t](#) *con)
Gracefully destroy a POP3 session, whether because of an error or in response to a user QUIT command.
- void [pop_user](#) ([connection_t](#) *con)
Accept a username for POP3 authentication.
- void [pop_pass](#) ([connection_t](#) *con)
Accept and verify a password for POP3 authentication.
- void [pop_capa](#) ([connection_t](#) *con)
Display the POP3 server capabilities, in response to a POP3 CAPA command.
- void [pop_stat](#) ([connection_t](#) *con)
Display a user's message statistics, in response to a POP3 STAT command.
- void [pop_last](#) ([connection_t](#) *con)
Get the sequence number of the last read message, in response to a POP3 LAST command.
- void [pop_list](#) ([connection_t](#) *con)
Get the list of a user's messages, in response to a POP3 LIST command.
- void [pop_dele](#) ([connection_t](#) *con)
Get a message, in response to a POP3 DELE command.
- void [pop_uidl](#) ([connection_t](#) *con)
Get the UIDL for a message or collection of messages, in response to a POP3 UIDL command.
- void [pop_top](#) ([connection_t](#) *con)
Get the top lines of a message or collection of messages, in response to a POP3 TOP command.
- void [pop_retr](#) ([connection_t](#) *con)
Retrieve a user's message, in response to a POP3 RETR command.
- void [pop_init](#) ([connection_t](#) *con)
Initialize a new POP3 connection.

5.437.1 Function Documentation

5.437.1.1 void pop_capa (connection_t * con)

Display the POP3 server capabilities, in response to a POP3 CAPA command. [pop.c](#)

Note:

See RFC 2449 POP3 Extension Mechanism for details.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 342 of file pop.c.

References [build_version\(\)](#), [con_print\(\)](#), and [con_secure\(\)](#).

5.437.1.2 void pop_delete (connection_t * con)

Get a message, in response to a POP3 DELE command.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 478 of file pop.c.

References [con_write_bl\(\)](#), [MAIL_STATUS_HIDDEN](#), [meta_message_t](#), [meta_user_unlock\(\)](#), [meta_user_wlock\(\)](#), [number](#), [pop_get_message\(\)](#), [pop_invalid\(\)](#), and [pop_num_parse\(\)](#).

5.437.1.3 void pop_init (connection_t * con)

Initialize a new POP3 connection.

Parameters:

con the newly connected POP3 client connection.

Returns:

This function returns no value.

Definition at line 723 of file pop.c.

References [con_write_bl\(\)](#), and [pop_requeue\(\)](#).

Referenced by [protocol_enqueue\(\)](#).

5.437.1.4 void pop_invalid (connection_t * con)

A function handler for invalid POP3 commands.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 67 of file pop.c.

References con_write_bl().

Referenced by pop_dele(), pop_last(), pop_list(), pop_pass(), pop_process(), pop_retr(), pop_rset(), pop_starttls(), pop_stat(), pop_top(), pop_uidl(), and pop_user().

5.437.1.5 void pop_last (connection_t * con)

Get the sequence number of the last read message, in response to a POP3 LAST command.

See also:

[pop_get_last\(\)](#)

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 390 of file pop.c.

References con_print(), meta_user_rlock(), meta_user_unlock(), number, pop_get_last(), and pop_invalid().

5.437.1.6 void pop_list (connection_t * con)

Get the list of a user's messages, in response to a POP3 LIST command.

See also:

[pop_total_messages\(\)](#), [pop_get_message\(\)](#)

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 415 of file pop.c.

References con_print(), con_write_bl(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MAIL_STATUS_APPENDED, MAIL_STATUS_HIDDEN, meta_message_t, meta_user_rlock(), meta_user_unlock(), number, pop_get_message(), pop_invalid(), pop_num_parse(), and pop_total_messages().

5.437.1.7 void pop_noop (connection_t * con)

Execute a POP3 no-operation command.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 56 of file pop.c.

References `con_write_bl()`.

5.437.1.8 void pop_pass (connection_t * con)

Accept and verify a password for POP3 authentication.

Note:

This command is only allowed for sessions which have not yet been authenticated, but which have already supplied a username. If the username/password combo was validated, the account information is retrieved and checked to see if it is locked. After successful authentication, this function will prohibit insecure connections for any user configured to use TLS only, and enforce the existence of only one POP3 session at a time. Finally, the database Log table for this user's POP3 access is updated, and all the user's messages are retrieved.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 168 of file pop.c.

References `auth_free()`, `AUTH_LOCK_ABUSE`, `AUTH_LOCK_ADMIN`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_USER`, `auth_login()`, `cache_decrement()`, `cache_increment()`, `con_addr_presentation()`, `con_addr_subnet()`, `con_secure()`, `con_write_bl()`, `inx_count()`, `auth_t::keys`, `lock_get()`, `lock_release()`, `auth_t::locked`, `log_info`, `MANAGEDBUF`, `auth_t::master`, `meta_data_update_log()`, `meta_get()`, `META_GET_KEYS`, `META_GET_MESSAGES`, `meta_inx_remove()`, `META_LOCKED`, `meta_messages_login_update()`, `META_PROTOCOL_POP`, `META_USER_ENCRYPT_DATA`, `meta_user_ref_protocol_total()`, `META_USER_TLS`, `meta_user_unlock()`, `meta_user_wlock()`, `pop_invalid()`, `pop_pass_parse()`, `auth_t::salt`, `auth_t::seasoning`, `st_char_get()`, `st_dupe()`, `st_empty`, `st_free()`, `st_length_int()`, `st_populated`, `st_quick()`, `auth_t::status`, `time_datestamp()`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, and `auth_t::verification`.

5.437.1.9 void pop_quit (connection_t * con)

Gracefully destroy a POP3 session, whether because of an error or in response to a user QUIT command.

Parameters:

con the POP3 client connection to be shut down. This function returns no value.

Definition at line 107 of file pop.c.

References `con_destroy()`, `con_status()`, and `con_write_bl()`.

Referenced by `pop_process()`, and `pop_queue()`.

5.437.1.10 void pop_retr (connection_t * con)

Retrieve a user's message, in response to a POP3 RETR command.

Note:

This function will fail if a deleted message was specified by the user.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 653 of file pop.c.

References con_print(), con_write_bl(), con_write_st(), mail_destroy(), mail_load_message(), MAIL_STATUS_HIDDEN, meta_message_t, meta_user_flock(), meta_user_unlock(), number, PLACER, pop_get_message(), pop_invalid(), pop_num_parse(), st_char_get(), st_length_get(), st_replace(), and mail_message_t::text.

5.437.1.11 void pop_rset (connection_t * con)

Reset the user's mailbox, in response to a POP3 RSET command.

See also:

[pop_session_reset\(\)](#)

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 82 of file pop.c.

References con_write_bl(), pop_invalid(), and pop_session_reset().

5.437.1.12 void pop_starttls (connection_t * con)

TODO: Review error messages and update them with the appropriate response code, as per RFC 3206 regarding the response codes extension. Initialize a TLS session for an unauthenticated POP3 session.

Note:

RFC 2595 / section 4 dictates that the STLS/STARTTLS command should only be available in the authorization state.

Parameters:

con the connection of the POP3 client requesting the transport layer security upgrade.

Returns:

This function returns no value (all error messages are written directly to the requesting client).

Definition at line 18 of file pop.c.

References con_secure(), con_write_bl(), log_pedantic, M_SSL_BIO_NOCLOSE, pl_null(), pop_invalid(), pop_session_reset(), st_length_set(), stats_increment_by_name(), and tls_server_alloc().

5.437.1.13 void pop_stat (connection_t * con)

Display a user's message statistics, in response to a POP3 STAT command.

See also:

[pop_total_messages\(\)](#), [pop_total_size\(\)](#)

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 364 of file pop.c.

References [con_print\(\)](#), [count](#), [meta_user_rlock\(\)](#), [meta_user_unlock\(\)](#), [pop_invalid\(\)](#), [pop_total_messages\(\)](#), and [pop_total_size\(\)](#).

5.437.1.14 void pop_top (connection_t * con)

Get the top lines of a message or collection of messages, in response to a POP3 TOP command.

Note:

This function will fail if a deleted message was specified by the user.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 582 of file pop.c.

References [con_print\(\)](#), [con_write_bl\(\)](#), [con_write_st\(\)](#), [mail_destroy\(\)](#), [mail_load_message_top\(\)](#), [MAIL_STATUS_HIDDEN](#), [meta_message_t](#), [meta_user_rlock\(\)](#), [meta_user_unlock\(\)](#), [number](#), [PLACER](#), [pop_get_message\(\)](#), [pop_invalid\(\)](#), [pop_top_parse\(\)](#), [st_char_get\(\)](#), [st_length_get\(\)](#), [st_replace\(\)](#), and [mail_message_t::text](#).

5.437.1.15 void pop_uidl (connection_t * con)

Get the UIDL for a message or collection of messages, in response to a POP3 UIDL command.

Parameters:

con the POP3 client connection issuing the command.

Returns:

This function returns no value.

Definition at line 517 of file pop.c.

References [con_print\(\)](#), [con_write_bl\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [MAIL_STATUS_APPENDED](#), [MAIL_STATUS_HIDDEN](#), [meta_message_t](#), [meta_user_rlock\(\)](#), [meta_user_unlock\(\)](#), [number](#), [pop_get_message\(\)](#), [pop_invalid\(\)](#), [pop_num_parse\(\)](#), and [pop_total_messages\(\)](#).

5.437.1.16 void pop_user (connection_t * *con*)

Accept a username for POP3 authentication.

Note:

This command is only allowed for sessions which have not yet been authenticated. If the username has already been supplied pre-authentication, the old value will be overwritten with the new one.

Parameters:

con the POP3 client connection issuing the command. This function returns no value.

Definition at line 131 of file pop.c.

References `con_write_bl()`, `pop_invalid()`, `pop_user_parse()`, and `st_cleanup`.

5.438 src/servers/smtp/accept.c File Reference

```
#include "magma.h"
```

Functions

- [int_t smtp_store_message](#) ([smtp_inbound_prefs_t](#) *prefs, [stringer_t](#) **local)
Store a received SMTP message as a generic mail message, both on disk and in the database.
- [int_t smtp_rollout](#) ([smtp_inbound_prefs_t](#) *prefs)
Delete the oldest mail message owned by a user until their storage usage falls below their storage quota.
- [bool_t smtp_store_spamsig](#) ([smtp_inbound_prefs_t](#) *prefs, [int_t](#) spam)
Generate a random key for a spam signature and store it in the database.
- [int_t smtp_accept_message](#) ([connection_t](#) *con, [smtp_inbound_prefs_t](#) *prefs)
accept.c

5.438.1 Function Documentation

5.438.1.1 [int_t smtp_accept_message](#) ([connection_t](#) *con, [smtp_inbound_prefs_t](#) *prefs)

[accept.c](#)

Definition at line 185 of file [accept.c](#).

References [magma_t::abuse](#), [magma_t::admin](#), [smtp_inbound_prefs_t::autoreply](#), [smtp_inbound_prefs_t::bounces](#), [magma_t::contact](#), [smtp_inbound_prefs_t::dkim](#), [dkim_signature_verify\(\)](#), [smtp_inbound_prefs_t::dkimaction](#), [dspam_check\(\)](#), [smtp_inbound_prefs_t::filters](#), [smtp_inbound_prefs_t::foldernum](#), [smtp_inbound_prefs_t::forwarded](#), [smtp_inbound_prefs_t::inbox](#), [log_error](#), [log_pedantic](#), [magma](#), [mail_add_inbound_headers\(\)](#), [smtp_inbound_prefs_t::mark](#), [smtp_inbound_prefs_t::overquota](#), [smtp_inbound_prefs_t::phish](#), [smtp_inbound_prefs_t::phishaction](#), [PLACER](#), [smtp_inbound_prefs_t::rbl](#), [smtp_inbound_prefs_t::rblaction](#), [smtp_inbound_prefs_t::rcptto](#), [smtp_inbound_prefs_t::rcv_size_limit](#), [smtp_inbound_prefs_t::rollout](#), [smtp_inbound_prefs_t::signum](#), [SMTP_ACTION_BOUNCE](#), [SMTP_ACTION_DELETE](#), [SMTP_ACTION_MARK](#), [SMTP_ACTION_MARK_READ](#), [smtp_check_filters\(\)](#), [smtp_forward_message\(\)](#), [SMTP_MARK_NONE](#), [SMTP_MARK_PHISH](#), [SMTP_MARK_RBL](#), [SMTP_MARK_READ](#), [SMTP_MARK_SPAM](#), [SMTP_MARK_SPOOF](#), [SMTP_MARK_VIRUS](#), [SMTP_OUTCOME_BOUNCE_DKIM](#), [SMTP_OUTCOME_BOUNCE_PHISH](#), [SMTP_OUTCOME_BOUNCE_RBL](#), [SMTP_OUTCOME_BOUNCE_SPAM](#), [SMTP_OUTCOME_BOUNCE_SPF](#), [SMTP_OUTCOME_BOUNCE_VIRUS](#), [SMTP_OUTCOME_PERM_FAILURE](#), [SMTP_OUTCOME_SUCESS](#), [SMTP_OUTCOME_TEMP_LOCKED](#), [SMTP_OUTCOME_TEMP_OVERQUOTA](#), [SMTP_OUTCOME_TEMP_SERVER](#), [smtp_reply\(\)](#), [smtp_rollout\(\)](#), [smtp_store_message\(\)](#), [smtp_store_spamsig\(\)](#), [smtp_update_receive_stats\(\)](#), [smtp_inbound_prefs_t::spam](#), [smtp_inbound_prefs_t::spam_checked](#), [smtp_inbound_prefs_t::spamaction](#), [smtp_inbound_prefs_t::spamkey](#), [smtp_inbound_prefs_t::spamsig](#), [smtp_inbound_prefs_t::spf](#), [smtp_inbound_prefs_t::spfaction](#), [st_cmp_ci_eq\(\)](#), [st_cmp_cs_eq\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [smtp_inbound_prefs_t::usernum](#), [smtp_inbound_prefs_t::virus](#), [virus_check\(\)](#), and [smtp_inbound_prefs_t::virusaction](#).

Referenced by [smtp_data_inbound\(\)](#).

5.438.1.2 [int_t smtp_rollout](#) ([smtp_inbound_prefs_t](#) *prefs)

Delete the oldest mail message owned by a user until their storage usage falls below their storage quota.

Parameters:

prefs a pointer to the specified user's inbound mail preferences data.

Returns:

1 on success or < 0 on failure, where -1: An error occurred retrieving the rollout message list from the database. -2: The user lock could not be acquired.

Definition at line 79 of file accept.c.

References `log_pedantic`, `mail_remove_message()`, `OBJECT_MESSAGES`, `smtp_inbound_prefs_t::quota`, `res_field_string()`, `res_field_uint32()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `serial_increment()`, `smtp_fetch_rollmessages()`, `st_char_get()`, `st_free()`, `smtp_inbound_prefs_t::stor_size`, `user_lock()`, `user_unlock()`, and `smtp_inbound_prefs_t::usernum`.

Referenced by `smtp_accept_message()`.

5.438.1.3 `int_t smtp_store_message (smtp_inbound_prefs_t * prefs, stringer_t ** local)`

Store a received SMTP message as a generic mail message, both on disk and in the database.

See also:

`mail_store_messages()`

Returns:

-1 on failure or 1 on success.

Definition at line 15 of file accept.c.

References `smtp_inbound_prefs_t::foldernum`, `log_error`, `log_pedantic`, `MAIL_MARK_BLACKHOLED`, `MAIL_MARK_INFECTED`, `MAIL_MARK_JUNK`, `MAIL_MARK_PHISHING`, `MAIL_MARK_SPOOFED`, `MAIL_STATUS_RECENT`, `MAIL_STATUS_SEEN`, `mail_store_message()`, `smtp_inbound_prefs_t::mark`, `smtp_inbound_prefs_t::messagenum`, `OBJECT_MESSAGES`, `serial_increment()`, `smtp_inbound_prefs_t::signet`, `smtp_inbound_prefs_t::signum`, `SMTP_MARK_PHISH`, `SMTP_MARK_RBL`, `SMTP_MARK_READ`, `SMTP_MARK_SPAM`, `SMTP_MARK_SPOOF`, `SMTP_MARK_VIRUS`, `smtp_inbound_prefs_t::spamkey`, `status`, `user_lock()`, `user_unlock()`, and `smtp_inbound_prefs_t::usernum`.

Referenced by `smtp_accept_message()`.

5.438.1.4 `bool_t smtp_store_spamsig (smtp_inbound_prefs_t * prefs, int_t spam)`

Generate a random key for a spam signature and store it in the database.

Parameters:

prefs the user's smtp inbound preferences object, with the spam signature field set.

spam the dspam return code associated with the spam signature.

Returns:

true if the key was inserted into the database successfully, or false on failure.

Definition at line 161 of file accept.c.

References `imap_fetch_response_t::key`, `log_pedantic`, `rand_get_uint64()`, `smtp_inbound_prefs_t::signum`, `smtp_insert_spamsig()`, `smtp_inbound_prefs_t::spamkey`, and `uint64_digits()`.

Referenced by `smtp_accept_message()`.

5.439 src/servers/smtp/checkers.c File Reference

```
#include "magma.h"
```

Functions

- [int_t smtp_check_greylis](#)([connection_t](#) *con, [smtp_inbound_prefs_t](#) *prefs)
Check to see if a transmitting address is in a user's greylis.
- [int_t smtp_check_rbl](#)([connection_t](#) *con)
Check the SMTP connection's remote address against a collection of real-time blacklists.
- [int_t smtp_check_filters](#)([smtp_inbound_prefs_t](#) *prefs, [stringer_t](#) **local)
checkers.c
- [bool_t smtp_add_bypass_entry](#)([stringer_t](#) *subnet)
Add an entry to the SMTP subnet bypass list.
- [bool_t smtp_bypass_check](#)([connection_t](#) *con)
Check if a connection should bypass certain SMTP checks.

5.439.1 Function Documentation

5.439.1.1 [bool_t smtp_add_bypass_entry](#)([stringer_t](#) * *subnet*)

Add an entry to the SMTP subnet bypass list.

Parameters:

subnet a pointer to a managed string containing the IP address or subnet address to be bypassed for checks.

Returns:

true if the entry was valid and was added, or false on failure.

Definition at line 249 of file checkers.c.

References [magma_t::bypass_subnets](#), [inx_alloc\(\)](#), [inx_insert\(\)](#), [ip_subnet_st\(\)](#), [log_error](#), [log_pedantic](#), [M_INX_LINKED](#), [M_TYPE_UINT64](#), [magma](#), [mm_alloc\(\)](#), [mm_free\(\)](#), [magma_t::smtp](#), [st_char_get\(\)](#), [multi_t::type](#), [multi_t::u64](#), and [multi_t::val](#).

Referenced by [config_value_set\(\)](#).

5.439.1.2 [bool_t smtp_bypass_check](#)([connection_t](#) * *con*)

Check if a connection should bypass certain SMTP checks.

Note:

This check is run against host and/or subnet masks configured in the `magma.smtp.bypass_addr` option.

Parameters:

con a pointer to the connection object to be checked.

Returns:

true if the specified connection meets the SMTP bypass check or false on failure or if it does not.

Definition at line 289 of file checkers.c.

References magma_t::bypass_subnets, con_addr(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), ip_matches_subnet(), magma, and magma_t::smtp.

Referenced by smtp_init().

5.439.1.3 int_t smtp_check_filters (smtp_inbound_prefs_t * prefs, stringer_t ** local)

[checkers.c](#) Apply any user specific filters. Return -1 for errors, and -2 to delete a message. Return 1 if no action was taken, and 2 if the message was moved to a different folder, 3 if the message content was modified, and 4 if the message was marked read.

Definition at line 126 of file checkers.c.

References data, smtp_inbound_prefs_t::filters, smtp_inbound_prefs_t::foldernum, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), length, log_pedantic, mail_header_end(), mail_header_fetch_all(), mail_mod_subject(), smtp_inbound_prefs_t::mark, MEMORYBUF, mm_wipe(), pl_data_get(), pl_init(), pl_length_get(), pl_null(), PLACER, placer_t, SMTP_FILTER_ACTION_DELETE, SMTP_FILTER_ACTION_LABEL, SMTP_FILTER_ACTION_MARK_READ, SMTP_FILTER_ACTION_MOVE, SMTP_FILTER_LOCATION_BODY, SMTP_FILTER_LOCATION_ENTIRE, SMTP_FILTER_LOCATION_FIELD, SMTP_FILTER_LOCATION_HEADER, SMTP_MARK_NONE, SMTP_MARK_PHISH, SMTP_MARK_RBL, SMTP_MARK_READ, SMTP_MARK_SPAM, SMTP_MARK_SPOOF, SMTP_MARK_VIRUS, st_char_get(), st_cleanup, st_cmp_ci_eq(), st_length_get(), st_length_int(), and smtp_inbound_prefs_t::usernum.

Referenced by smtp_accept_message().

5.439.1.4 int_t smtp_check_greylist (connection_t * con, smtp_inbound_prefs_t * prefs)

Check to see if a transmitting address is in a user's greylist.

Note:

The greylist is configured in the Dispatch table and specifies the minimum time, in minutes, that a transmitting smtp relay server must wait in order to be able to send more messages to the same recipient address again.

Parameters:

con the connection to have its remote address checked against the user's greylist.

prefs the smtp inbound preferences of the user

Returns:

-1 on an internal server error, 0 if the sender must wait longer, and 1 if the check was passed.

Definition at line 18 of file checkers.c.

References BLOCK_T, cache_get(), cache_set(), con_addr_reversed(), CONTIGUOUS, smtp_inbound_prefs_t::greytime, HEAP, imap_fetch_response_t::key, log_pedantic, MANAGEDBUF, st_alloc_opts(), st_char_get(), st_cleanup, st_data_get(), st_length_get(), st_length_int(), st_sprint(), smtp_inbound_prefs_t::usernum, and imap_fetch_response_t::value.

Referenced by smtp_rcpt_to().

5.439.1.5 int_t smtp_check_rbl (connection_t * con)

Check the SMTP connection's remote address against a collection of real-time blacklists.

Note:

The connection's IP address will be checked against each of the servers configured in magma.smtp.blacklists.domain.

Parameters:

con the connection to have its address examined against the RBLs.

Returns:

-1 on general error, -2 if the address was blacklisted, or 1 if it passed the check.

Definition at line 77 of file checkers.c.

References magma_t::blacklists, con_addr_reversed(), log_pedantic, magma, MANAGEDBUF, mm_wipe(), magma_t::smtp, st_char_get(), and st_length_int().

Referenced by smtp_rcpt_to().

5.440 src/servers/smtp/smtp.c File Reference

```
#include "magma.h"
```

Functions

- void [smtp_starttls](#) ([connection_t](#) *con)
TODO: Review error messages and update them with the appropriate response code.
- void [smtp_mail_from](#) ([connection_t](#) *con)
Specify the identity of a message's sender, in response to an SMTP MAIL FROM command.
- void [smtp_ehlo](#) ([connection_t](#) *con)
Process an SMTP EHLO command.
- void [smtp_helo](#) ([connection_t](#) *con)
Process an SMTP HELO command.
- void [smtp_noop](#) ([connection_t](#) *con)
Perform an SMTP NOOP (no-operation) command.
- void [smtp_disabled](#) ([connection_t](#) *con)
A stub function for an SMTP command that has not been implemented.
- void [smtp_invalid](#) ([connection_t](#) *con)
A function that is executed when an invalid SMTP command is executed.
- void [smtp_quit](#) ([connection_t](#) *con)
Gracefully terminate an SMTP session, especially in response to an SMTP QUIT command.
- void [smtp_rset](#) ([connection_t](#) *con)
Reset the SMTP session, in response to an SMTP RSET command.
- void [smtp_auth_plain](#) ([connection_t](#) *con)
- void [smtp_auth_login](#) ([connection_t](#) *con)
smtp.c
- void [smtp_rcpt_to](#) ([connection_t](#) *con)
- void [smtp_data_finish](#) ([connection_t](#) *con, [size_t](#) read, [int_t](#) checker)
- [int_t](#) [smtp_data_read](#) ([connection_t](#) *con, [stringer_t](#) **message)
- void [smtp_data_outbound](#) ([connection_t](#) *con)
- void [smtp_data_inbound](#) ([connection_t](#) *con)
- void [smtp_data](#) ([connection_t](#) *con)
- void [smtp_init](#) ([connection_t](#) *con)
The start of the protocol handler for the SMTP server.
- void [submission_init](#) ([connection_t](#) *con)

5.440.1 Function Documentation

5.440.1.1 void smtp_auth_login (connection_t * con)

smtp.c

Definition at line 393 of file smtp.c.

References `auth_free()`, `AUTH_LOCK_ABUSE`, `AUTH_LOCK_ADMIN`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_USER`, `auth_login()`, `base64_decode()`, `cache_decrement()`, `cache_increment()`, `con_addr_presentation()`, `con_addr_subnet()`, `con_read_line()`, `con_write_bl()`, `lock_get()`, `lock_release()`, `auth_t::locked`, `log_info`, `MANAGEDBUF`, `pl_char_get()`, `pl_length_get()`, `PLACER`, `smtp_outbound_prefs_t::send_size_limit`, `smtp_add_outbound()`, `smtp_fetch_authorization()`, `smtp_parse_auth()`, `smtp_session_reset()`, `st_char_get()`, `st_cleanup`, `st_cmp_ci_eq()`, `st_empty`, `st_free()`, `st_length_int()`, `st_populated`, `st_quick()`, `auth_t::status`, `time_datestamp()`, `auth_t::tokens`, `auth_t::username`, and `auth_t::verification`.

5.440.1.2 void smtp_auth_plain (connection_t * con)

Definition at line 224 of file smtp.c.

References `auth_free()`, `AUTH_LOCK_ABUSE`, `AUTH_LOCK_ADMIN`, `AUTH_LOCK_EXPIRED`, `AUTH_LOCK_USER`, `auth_login()`, `base64_decode()`, `cache_decrement()`, `cache_increment()`, `con_addr_presentation()`, `con_addr_subnet()`, `con_read_line()`, `con_write_bl()`, `FOREIGNDATA`, `JOINTED`, `lock_get()`, `lock_release()`, `auth_t::locked`, `log_info`, `MANAGEDBUF`, `mm_copy()`, `pl_char_get()`, `pl_length_get()`, `PLACER`, `PLACER_T`, `placer_t`, `smtp_outbound_prefs_t::send_size_limit`, `smtp_add_outbound()`, `smtp_fetch_authorization()`, `smtp_parse_auth()`, `smtp_session_reset()`, `st_char_get()`, `st_cleanup`, `st_cmp_ci_eq()`, `st_empty`, `st_free()`, `st_length_get()`, `st_length_int()`, `st_populated`, `st_quick()`, `STACK`, `auth_t::status`, `time_datestamp()`, `tok_get_st()`, `auth_t::tokens`, `auth_t::username`, and `auth_t::verification`.

5.440.1.3 void smtp_data (connection_t * con)

Definition at line 1149 of file smtp.c.

References `con_print()`, `con_write_bl()`, `magma`, `mail_add_required_headers()`, `mail_count_received()`, `mail_create_message()`, `mail_destroy_message()`, `mail_message_cleanup()`, `magma_t::relay_limit`, `requeue()`, `magma_t::smtp`, `smtp_data_inbound()`, `smtp_data_outbound()`, `smtp_data_read()`, `smtp_quit()`, `smtp_requeue()`, and `st_free()`.

Referenced by `smtp_process()`.

5.440.1.4 void smtp_data_finish (connection_t * con, size_t read, int_t checker)

Definition at line 803 of file smtp.c.

References `con_read()`, `st_char_get()`, `st_data_get()`, `st_data_set()`, `st_length_set()`, and `status`.

Referenced by `smtp_data_read()`.

5.440.1.5 void smtp_data_inbound (connection_t * con)

Definition at line 1101 of file smtp.c.

References `con_print()`, `con_write_bl()`, `smtp_inbound_prefs_t::next`, `smtp_inbound_prefs_t::outcome`, `smtp_accept_message()`, `smtp_bounce()`, `SMTP_OUTCOME_PERM_FAILURE`, `SMTP_OUTCOME_SUCESS`, `SMTP_OUTCOME_TEMP_LOCKED`, `SMTP_OUTCOME_TEMP_OVERQUOTA`, `SMTP_OUTCOME_TEMP_SERVER`, and `smtp_session_reset()`.

Referenced by `smtp_data()`.

5.440.1.6 void smtp_data_outbound (connection_t * con)

Definition at line 1008 of file smtp.c.

References `auth_sanitize_address()`, `con_print()`, `con_write_bl()`, `con_write_st()`, `mail_extract_address()`, `pattern_check()`, `PLACER`, `smtp_check_authorized_from()`, `smtp_check_transmit_quota()`, `smtp_relay_message()`, `smtp_session_reset()`, `smtp_update_transmission_stats()`, `st_char_get()`, `st_cleanup`, `st_cmp_ci_eq()`, `st_dupe()`, `st_free()`, `st_length_get()`, and `virus_check()`.

Referenced by `smtp_data()`.

5.440.1.7 `int_t smtp_data_read (connection_t * con, stringer_t ** message)`

Definition at line 859 of file `smtp.c`.

References `con_read()`, `HEAP`, `JOINTED`, `log_pedantic`, `MAPPED_T`, `smtp_data_finish()`, `st_alloc_opts()`, `st_char_get()`, `st_data_get()`, `st_data_set()`, `st_free()`, `st_length_set()`, `st_realloc()`, and `status`.

Referenced by `smtp_data()`.

5.440.1.8 `void smtp_disabled (connection_t * con)`

A stub function for an SMTP command that has not been implemented.

Note:

Executing a disabled command while result in a small delay and the protocol violation counter being incremented.

Returns:

This function returns no value.

Definition at line 163 of file `smtp.c`.

References `con_print()`.

5.440.1.9 `void smtp_ehlo (connection_t * con)`

Process an SMTP EHLO command.

See also:

[smtp_parse_helo_domain\(\)](#)

Note:

Any prior domain specified by a HELO/EHLO command will be overwritten.

Parameters:

con the SMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 98 of file `smtp.c`.

References `con_print()`, `con_secure()`, `con_write_bl()`, `magma`, `magma_t::message_length_limit`, `magma_t::smtp`, `smtp_parse_helo_domain()`, `st_char_get()`, `st_cleanup`, and `st_length_int()`.

5.440.1.10 `void smtp_helo (connection_t * con)`

Process an SMTP HELO command.

See also:

[smtp_parse_helo_domain\(\)](#)

Note:

Any prior domain specified by a HELO/EHLO command will be overwritten.

Parameters:

con the SMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 127 of file smtp.c.

References [con_print\(\)](#), [con_write_bl\(\)](#), [smtp_parse_helo_domain\(\)](#), [st_char_get\(\)](#), [st_cleanup](#), and [st_length_int\(\)](#).

5.440.1.11 void smtp_init (connection_t * con)

The start of the protocol handler for the SMTP server.

Parameters:

con the new inbound SMTP client connection.

Returns:

This function returns no value.

Definition at line 1260 of file smtp.c.

References [con_print\(\)](#), [con_reverse_enqueue\(\)](#), [smtp_bypass_check\(\)](#), [smtp_requeue\(\)](#), [st_char_get\(\)](#), and [st_length_int\(\)](#).

Referenced by [protocol_enqueue\(\)](#), and [submission_init\(\)](#).

5.440.1.12 void smtp_invalid (connection_t * con)

A function that is executed when an invalid SMTP command is executed.

Returns:

This function returns no value.

Definition at line 176 of file smtp.c.

References [con_write_bl\(\)](#).

Referenced by [smtp_process\(\)](#).

5.440.1.13 void smtp_mail_from (connection_t * con)

Specify the identity of a message's sender, in response to an SMTP MAIL FROM command.

See also:

[smtp_parse_mail_from_path\(\)](#)

Note:

This command must be preceded by a HELO command and successful authentication. Any prior email address specified by a MAIL FROM command will be overwritten. If the SIZE parameter was specified with the MAIL FROM command, its value will be compared to the maximum value specified in the smtp.message_length_limit configuration option.

Parameters:

con the SMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 57 of file smtp.c.

References con_write_bl(), magma, magma_t::message_length_limit, magma_t::smtp, smtp_parse_mail_from_path(), smtp_session_reset(), and st_free().

5.440.1.14 void smtp_noop (connection_t * con)

Perform an SMTP NOOP (no-operation) command.

Note:

This command does essentially nothing and is mostly a way to keep connections alive without timing out due to inactivity.

Returns:

This function returns no value.

Definition at line 152 of file smtp.c.

References con_write_bl().

5.440.1.15 void smtp_quit (connection_t * con)

Gracefully terminate an SMTP session, especially in response to an SMTP QUIT command.

Note:

The standards specify that the receiver MUST send an OK reply, and then close the transmission channel.

Parameters:

the SMTP client connection to be terminated.

Returns:

This function returns no value.

Definition at line 191 of file smtp.c.

References con_destroy(), con_flush(), con_status(), and con_write_bl().

Referenced by smtp_data(), smtp_process(), and smtp_requeue().

5.440.1.16 void smtp_rcpt_to (connection_t * con)

BUG: Detect messages 'from' a local user/domain and tell them to authenticate first.

Definition at line 550 of file smtp.c.

References magma_t::abuse, magma_t::admin, AUTH_LOCK_ABUSE, AUTH_LOCK_ADMIN, AUTH_LOCK_EXPIRED, AUTH_LOCK_INACTIVITY, AUTH_LOCK_USER, auth_sanitize_address(), con_addr(), con_addr_presentation(), con_print(), con_secure(), con_write_bl(), magma_t::contact, smtp_inbound_prefs_t::daily_recv_limit, smtp_inbound_prefs_t::daily_recv_limit_ip, smtp_inbound_prefs_t::forwarded, smtp_inbound_prefs_t::greylist, smtp_inbound_prefs_t::greytime, lower_st(), magma, mail_domain_get(), MANAGED-BUF, MEMORYBUF, smtp_inbound_prefs_t::overquota, pl_null(), placer_t, smtp_inbound_prefs_t::rbl, smtp_inbound_prefs_t::rblaction, smtp_inbound_prefs_t::rcptto, magma_t::recipient_limit, smtp_inbound_prefs_t::rcv_size_limit, smtp_inbound_prefs_t::rollout, magma_t::smtp, SMTP_ACTION_REJECT, smtp_add_inbound(), smtp_add_recipient(), smtp_check_duplicate_recipient(), smtp_check_greylist(), smtp_check_rbl(), smtp_check_receive_quota(), smtp_fetch_inbound(), smtp_free_inbound(), smtp_parse_rcpt_to(), smtp_inbound_prefs_t::spf, spf_check(), smtp_inbound_prefs_t::spfaction, st_char_get(), st_cmp_ci_eq(), st_free(), st_length_int(), and smtp_inbound_prefs_t::usernum.

5.440.1.17 void smtp_rset (connection_t * con)

Reset the SMTP session, in response to an SMTP RSET command.

Note:

This command clears any sender, recipient, and mail data, along with all buffers and state tables. The connection structure is reset to the same it was in immediately after the HELO/EHLO command.

Parameters:

con the SMTP client connection issuing the command.

Returns:

This function returns no value.

Definition at line 216 of file smtp.c.

References con_write_bl(), and smtp_session_reset().

5.440.1.18 void smtp_starttls (connection_t * con)

TODO: Review error messages and update them with the appropriate response code. Initialize a TLS session for an unauthenticated SMTP session.

Parameters:

con the connection of the SMTP endpoint requesting the transport layer security upgrade.

Returns:

This function returns no value.

Definition at line 17 of file smtp.c.

References con_secure(), con_write_bl(), log_pedantic, M_SSL_BIO_NOCLOSE, pl_null(), smtp_session_reset(), st_length_set(), stats_increment_by_name(), and tls_server_alloc().

5.440.1.19 void submission_init (connection_t * con)

Definition at line 1277 of file smtp.c.

References smtp_init().

Referenced by protocol_enqueue().

5.441 src/servers/smtp/transmit.c File Reference

```
#include "magma.h"
```

Functions

- [int_t smtp_relay_message](#) ([connection_t](#) *con, [stringer_t](#) **result)

Relay an outbound smtp message for a user.

- [int_t smtp_forward_message](#) ([server_t](#) *server, [stringer_t](#) *sender, [stringer_t](#) *address, [stringer_t](#) *message, [stringer_t](#) *id, [int_t](#) mark, [uint64_t](#) signum, [uint64_t](#) sigkey)
- [int_t smtp_bounce](#) ([connection_t](#) *con)

transmit.c

- [int_t smtp_reply](#) ([stringer_t](#) *from, [stringer_t](#) *to, [uint64_t](#) usernum, [uint64_t](#) autoreply, [int_t](#) spf, [int_t](#) dkim)
- [int_t smtp_send_message](#) ([stringer_t](#) *to, [stringer_t](#) *from, [stringer_t](#) *message)

Relay an outbound smtp message for the user.

5.441.1 Function Documentation

5.441.1.1 [int_t smtp_bounce](#) ([connection_t](#) * con)

[transmit.c](#)

Definition at line 179 of file [transmit.c](#).

References [client_t](#), [crc64_checksum\(\)](#), [dkim_signature_create\(\)](#), [magma_t::domain](#), [log_pedantic](#), [magma](#), [MANAGEDBUF](#), [smtp_inbound_prefs_t::next](#), [number](#), [smtp_inbound_prefs_t::outcome](#), [rand_choices\(\)](#), [smtp_inbound_prefs_t::rcptto](#), [smtp_client_close\(\)](#), [smtp_client_connect\(\)](#), [smtp_client_send_data\(\)](#), [smtp_client_send_helo\(\)](#), [smtp_client_send_nullfrom\(\)](#), [smtp_client_send_rcptto\(\)](#), [SMTP_OUTCOME_BOUNCE_DKIM](#), [SMTP_OUTCOME_BOUNCE_PHISH](#), [SMTP_OUTCOME_BOUNCE_RBL](#), [SMTP_OUTCOME_BOUNCE_SPAM](#), [SMTP_OUTCOME_BOUNCE_SPF](#), [SMTP_OUTCOME_BOUNCE_VIRUS](#), [SMTP_OUTCOME_PERM_FAILURE](#), [SMTP_OUTCOME_SUCESS](#), [SMTP_OUTCOME_TEMP_LOCKED](#), [SMTP_OUTCOME_TEMP_OVERQUOTA](#), [SMTP_OUTCOME_TEMP_SERVER](#), [st_char_get\(\)](#), [st_cleanup](#), [st_cmp_cs_eq\(\)](#), [st_free\(\)](#), [st_length_int\(\)](#), [st_merge](#), [st_sprint\(\)](#), and [magma_t::system](#).

Referenced by [smtp_data_inbound\(\)](#).

5.441.1.2 [int_t smtp_forward_message](#) ([server_t](#) * server, [stringer_t](#) * sender, [stringer_t](#) * address, [stringer_t](#) * message, [stringer_t](#) * id, [int_t](#) mark, [uint64_t](#) signum, [uint64_t](#) sigkey)

Definition at line 113 of file [transmit.c](#).

References [client_t](#), [HEAP](#), [JOINTED](#), [log_pedantic](#), [mail_add_forward_headers\(\)](#), [MAPPED_T](#), [PLACER](#), [smtp_client_close\(\)](#), [smtp_client_connect\(\)](#), [smtp_client_send_data\(\)](#), [smtp_client_send_helo\(\)](#), [smtp_client_send_mailfrom\(\)](#), [smtp_client_send_rcptto\(\)](#), [st_dupe_opts\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), and [st_replace\(\)](#).

Referenced by [smtp_accept_message\(\)](#).

5.441.1.3 [int_t smtp_relay_message](#) ([connection_t](#) * con, [stringer_t](#) ** result)

Relay an outbound smtp message for a user.

Note:

The following process occurs before the message will be sent: 1. Necessary outbound headers are attached to the message.* 2. An outbound connection to a mail relay server is established (with a premium or normal server pool). 3. Once the connection is negotiated,

an RCPT TO command is issued for each of the message's recipients. 4. The mail message data is sent and the client connection is closed.

Parameters:

con a pointer to the connection object across which the outbound mail was attempted to be sent.

result a pointer to the address of a managed string that will receive the server's last response to the mail send attempt, regardless of whether or not it was successful.

Returns:

1 if the message was successfully sent or -1 on failure.

Definition at line 22 of file transmit.c.

References smtp_recipients_t::address, client_t, CONTIGUOUS, HEAP, log_pedantic, mail_add_outbound_headers(), MANAGED_T, smtp_recipients_t::next, PLACER, smtp_client_close(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), smtp_client_send_rcptto(), st_dupe_opts(), and st_replace().

Referenced by smtp_data_outbound().

5.441.1.4 int_t smtp_reply (stringer_t *from, stringer_t *to, uint64_t usernum, uint64_t autoreply, int_t spf, int_t dkim)

Definition at line 363 of file transmit.c.

References cache_get_u64(), cache_set_u64(), client_t, crc64_checksum(), dkim_signature_create(), lock_get(), lock_release(), log_pedantic, MANAGEDBUF, rand_choices(), smtp_client_close(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_nullfrom(), smtp_client_send_rcptto(), smtp_fetch_autoreply(), st_char_get(), st_cleanup, st_free(), st_length_int(), st_merge, and st_sprint().

Referenced by smtp_accept_message().

5.441.1.5 int_t smtp_send_message (stringer_t *to, stringer_t *from, stringer_t *message)

Relay an outbound smtp message for the user.

Parameters:

to a managed string containing the name of the mail recipient.

from a managed string containing the address from which the email is being sent.

message a managed string containing the raw body of the mail message.

Returns:

-1 on error or 1 on success.

Definition at line 503 of file transmit.c.

References client_t, log_pedantic, smtp_client_close(), smtp_client_connect(), smtp_client_send_data(), smtp_client_send_helo(), smtp_client_send_mailfrom(), and smtp_client_send_rcptto().

Referenced by contact_business(), and register_business_step2().

5.442 src/web/contact/abuse.c File Reference

```
#include "magma.h"
```

Functions

- void `contact_abuse_increment_history` (`connection_t` *con)
Increment the contact abuse history counter for an IP address.
- `bool_t` `contact_abuse_checks` (`connection_t` *con, `chr_t` *branch)
Check to see that a client from a given IP address hasn't exceeded its daily quota of contact requests.

5.442.1 Function Documentation

5.442.1.1 `bool_t` `contact_abuse_checks` (`connection_t` * con, `chr_t` * branch)

Check to see that a client from a given IP address hasn't exceeded its daily quota of contact requests. abuse.c

Note:

Each IP address will be limited to at most 2 contact requests in any 24-hour period.

Parameters:

con a pointer to the connection object of the remote host making the contact request.

branch a null-terminated string specifying where the contact request was directed ("Abuse" or "Contact").

Returns:

true if the specified connection failed the abuse check or false if it did not.

Definition at line 34 of file abuse.c.

References `cache_get_u64()`, `con_addr_presentation()`, `contact_print_message()`, `MANAGEDBUF`, `st_char_get()`, `st_length_int()`, and `st_sprint()`.

Referenced by `contact_process()`.

5.442.1.2 `void` `contact_abuse_increment_history` (`connection_t` * con)

Increment the contact abuse history counter for an IP address.

Parameters:

con a pointer to the connection object of the remote host making the contact request.

Returns:

This function returns no value.

Definition at line 15 of file abuse.c.

References `cache_increment()`, `con_addr_presentation()`, `MANAGEDBUF`, `st_char_get()`, `st_length_int()`, and `st_sprint()`.

Referenced by `contact_business()`.

5.443 src/web/register/abuse.c File Reference

```
#include "magma.h"
```

Functions

- void [register_blocklist_free](#) (void)
Free a registration blocked list.
- void [register_blocklist_update](#) (void)
Update the registration blocklist from the database.
- [bool_t register_abuse_check_blocklist](#) ([connection_t](#) *con)
Check to see if the remote client is on the registration blocklist; if so, increment the web registration blocked counter.
- void [register_abuse_increment_history](#) ([connection_t](#) *con)
Increment the registration abuse counter for the requesting IP address.
- [bool_t register_abuse_checks](#) ([connection_t](#) *con)
Check to see if a registration request is allowed by a remote host; if not, display a banner.

Variables

- [inx_t](#) * [register_blocklist](#) = NULL
- [pthread_rwlock_t](#) [register_blocklist_lock](#) = PTHREAD_RWLOCK_INITIALIZER

5.443.1 Function Documentation

5.443.1.1 [bool_t register_abuse_check_blocklist](#) ([connection_t](#) * con)

Check to see if the remote client is on the registration blocklist; if so, increment the web registration blocked counter. abuse.c

Parameters:

con the client connection to be checked.

Returns:

false if the check was passed, or true if the connection was made from an IP on the blocklist.

Definition at line 58 of file abuse.c.

References [con_addr_standard\(\)](#), [inx_cursor_alloc\(\)](#), [inx_cursor_free\(\)](#), [inx_cursor_t](#), [inx_cursor_value_next\(\)](#), [MANAGEDBUF](#), [register_blocklist](#), [register_blocklist_lock](#), [rwlock_lock_read\(\)](#), [rwlock_unlock\(\)](#), [st_char_get\(\)](#), [st_cmp_ci_starts\(\)](#), [st_length_get\(\)](#), and [stats_increment_by_name\(\)](#).

Referenced by [register_abuse_checks\(\)](#).

5.443.1.2 [bool_t register_abuse_checks](#) ([connection_t](#) * con)

Check to see if a registration request is allowed by a remote host; if not, display a banner.

Parameters:

con the remote connection to be checked.

Returns:

false if registration is allowed or true if not (remote host is on blocklist or registration has been throttled).

Definition at line 112 of file abuse.c.

References `cache_get_u64()`, `con_addr_presentation()`, `con_addr_standard()`, `MANAGEDBUF`, `NULLER`, `register_abuse_check_blocklist()`, `register_print_message()`, and `st_char_get()`.

Referenced by `register_process()`.

5.443.1.3 void register_abuse_increment_history (connection_t * con)

Increment the registration abuse counter for the requesting IP address.

Parameters:

con a pointer to the connection object of the client making the registration request.

Returns:

This function returns no value.

Definition at line 95 of file abuse.c.

References `cache_increment()`, `con_addr_standard()`, `MANAGEDBUF`, `NULLER`, and `st_char_get()`.

Referenced by `api_endpoint_register()`, and `register_business_step2()`.

5.443.1.4 void register_blocklist_free (void)

Free a registration blocked list.

Returns:

This function returns no value.

Definition at line 18 of file abuse.c.

References `inx_free()`, and `register_blocklist`.

5.443.1.5 void register_blocklist_update (void)

Update the registration blocklist from the database.

Returns:

This function returns no value.

Definition at line 32 of file abuse.c.

References `inx_free()`, `inx_t`, `register_blocklist`, `register_blocklist_lock`, `register_data_fetch_blocklist()`, `rwlock_lock_write()`, and `rwlock_unlock()`.

5.443.2 Variable Documentation**5.443.2.1 inx_t* register_blocklist = NULL**

Definition at line 10 of file abuse.c.

Referenced by `register_abuse_check_blocklist()`, `register_blocklist_free()`, and `register_blocklist_update()`.

5.443.2.2 pthread_rwlock_t register_blocklist_lock = PTHREAD_RWLOCK_INITIALIZER

Definition at line 11 of file abuse.c.

Referenced by register_abuse_check_blocklist(), and register_blocklist_update().

5.444 src/web/contact/business.c File Reference

```
#include "magma.h"
```

Functions

- `http_page_t * contact_business_add_error (chr_t *branch, uchr_t *xpath, uchr_t *id, uchr_t *message)`
Return the contact/abuse page with a marked error indicator for the user in the event of a user submission error.
- `bool_t contact_business_valid_email (stringer_t *email)`
Validate an email address.
- `void contact_business (connection_t *con, chr_t *branch)`
Send the contents of a user submitted contact/abuse form to the configured contact email address.

5.444.1 Function Documentation

5.444.1.1 void contact_business (connection_t * con, chr_t * branch)

Send the contents of a user submitted contact/abuse form to the configured contact email address. business.c

Parameters:

- con** the connection of the web client making the request.
- branch** a null-terminated string containing the destination of the contact form: either "Abuse" or "Contact".

Returns:

This function returns no value.

Definition at line 127 of file business.c.

References magma_t::abuse, magma_t::admin, con_addr_presentation(), magma_t::contact, contact_abuse_increment_history(), contact_business_add_error(), contact_business_valid_email(), contact_print_form(), contact_print_message(), http_data_get(), HTTP_DATA_POST, log_error, magma, MANAGEDBUF, NULLER, PLACER, smtp_send_message(), st_cmp_cs_eq(), st_free(), st_merge, st_replace(), and http_data_t::value.

Referenced by contact_process().

5.444.1.2 http_page_t* contact_business_add_error (chr_t * branch, uchr_t * xpath, uchr_t * id, uchr_t * message)

Return the contact/abuse page with a marked error indicator for the user in the event of a user submission error.

Parameters:

- branch** a null-terminated string containing the source of the error: "Abuse" or "Contact"
- xpath** a null-terminated string containing the xpath of the element in the returned page that should be colored red.
- id** a null-terminated string containing the id of the error text element to be added to the document.
- message** a null-terminated string containing the actual error message text to be displayed to the user.

Returns:

NULL on failure, or a pointer to the processed contact page with error message on success.

Definition at line 18 of file business.c.

References `http_page_get()`, `NULLER`, `PLACER`, `st_cmp_cs_eq()`, `xml_node_add_sibling()`, `xml_node_free()`, `xml_node_new()`, `xml_node_set_content()`, `xml_node_set_property()`, `xml_xpath_eval()`, and `http_page_t::xpath_ctx`.

Referenced by `contact_business()`.

5.444.1.3 `bool_t contact_business_valid_email (stringer_t * email)`

Validate an email address.

Parameters:

email a managed string containing the email address to be validated.

Returns:

true if the specified email was valid and false if it was not.

Definition at line 60 of file business.c.

References `length`, `lower_st()`, `st_char_get()`, and `st_length_get()`.

Referenced by `config_validate_settings()`, and `contact_business()`.

5.445 src/web/register/business.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t register_business_validate_password](#) ([stringer_t](#) *password)
Determine whether a registered password is valid.
- [int_t register_business_validate_username](#) ([stringer_t](#) *username)
Determine whether a registered username is valid.
- [chr_t * register_business_step1](#) ([connection_t](#) *con, [register_session_t](#) *reg)
Perform verification checking on all step 1 completed user fields.
- [chr_t * register_business_step2](#) ([connection_t](#) *con, [register_session_t](#) *reg)
Perform verification checking on all step 2 completed user fields, and display a welcome banner on success.

5.445.1 Function Documentation

5.445.1.1 chr_t* register_business_step1 (connection_t * con, register_session_t * reg)

Perform verification checking on all step 1 completed user fields. business.c

Note:

Checks include captcha verification, username validation, and password reentry verification and validation.

Parameters:

con the underlying client connection.
reg the underlying registration session.

Returns:

NULL on success, or a descriptive error string on failure.

Definition at line 107 of file business.c.

References [data](#), [http_data_get\(\)](#), [HTTP_DATA_POST](#), [register_session_t::hvf_input](#), [register_session_t::hvf_value](#), [log_pedantic](#), [lower_st\(\)](#), [magma](#), [magma_t::minimum_password_length](#), [register_session_t::password](#), [register_business_validate_password\(\)](#), [register_business_validate_username\(\)](#), [register_data_check_username\(\)](#), [REGISTER_PASSWORD_MAX_LENGTH](#), [magma_t::secure](#), [st_char_get\(\)](#), [st_cmp_ci_eq\(\)](#), [st_dupe\(\)](#), [st_length_int\(\)](#), [register_session_t::username](#), and [http_data_t::value](#).

Referenced by [register_process\(\)](#).

5.445.1.2 chr_t* register_business_step2 (connection_t * con, register_session_t * reg)

Perform verification checking on all step 2 completed user fields, and display a welcome banner on success.

Note:

Checks include plan type validation, and billing information processing. After step 2, the user's supplied information will be persisted into the database.

Parameters:

con the underlying client connection.
reg the underlying registration session.

Returns:

NULL on success, or a descriptive error string on failure.

Definition at line 176 of file business.c.

References magma_t::admin, magma_t::contact, data, magma_t::domain, http_data_get(), HTTP_DATA_POST, log_pedantic, magma, register_session_t::password, PLACER, register_session_t::plan, register_abuse_increment_history(), register_data_insert_user(), smtp_send_message(), st_cleanup, st_cmp_cs_eq(), st_dupe(), st_merge, magma_t::system, tran_commit(), tran_rollback(), tran_start(), register_session_t::username, register_session_t::usernum, and http_data_t::value.

Referenced by register_process().

5.445.1.3 bool_t register_business_validate_password (stringer_t * password)

Determine whether a registered password is valid.

Note:

Each password must be between REGISTER_PASSWORD_MIN_LENGTH (5) and REGISTER_PASSWORD_MAX_LENGTH (200) characters long.

Parameters:

password the user's password to be evaluated.

Returns:

false on failure (too long, too short, or bad characters) or true on success.

Definition at line 16 of file business.c.

References length, magma, magma_t::minimum_password_length, REGISTER_PASSWORD_MAX_LENGTH, magma_t::secure, st_char_get(), st_length_get(), and utf8_length_st().

Referenced by register_business_step1().

5.445.1.4 int_t register_business_validate_username (stringer_t * username)

Determine whether a registered username is valid.

Parameters:

username a managed string containing the proposed username to be evaluated.

Returns:

-1 or 0 on failure (too long, too short, or bad characters) or 1 on success.

Definition at line 49 of file business.c.

References length, REGISTER_USERNAME_MAX_LENGTH, st_char_get(), and st_length_get().

Referenced by register_business_step1().

5.446 src/web/contact/contact.c File Reference

```
#include "magma.h"
```

Functions

- void [contact_print_message](#) ([connection_t](#) *con, [chr_t](#) *branch, [chr_t](#) *message)
Display the contact or abuse notification form to the requesting user.
- void [contact_print_form](#) ([connection_t](#) *con, [chr_t](#) *branch, [http_page_t](#) *page)
Display the contact or abuse form to the user.
- void [contact_process](#) ([connection_t](#) *con, [chr_t](#) *branch)
Process all user contact requests.

5.446.1 Function Documentation

5.446.1.1 void [contact_print_form](#) ([connection_t](#) *con, [chr_t](#) *branch, [http_page_t](#) *page)

Display the contact or abuse form to the user. [contact.c](#)

Parameters:

con the connection across which the form will be returned.

branch a null-terminated string specifying the type of contact, either "Contact" or "Abuse".

page a pointer to an http page containing the contact document to be populated with the user's name, email, and message.

Returns:

This function returns no value.

Definition at line 71 of file [contact.c](#).

References [con_write_st\(\)](#), [contact_print_message\(\)](#), [http_page_t::content](#), [data](#), [http_page_t::doc_obj](#), [http_data_get\(\)](#), [HTTP_DATA_POST](#), [http_page_free\(\)](#), [http_page_get\(\)](#), [http_response_header\(\)](#), [mm_cleanup](#), [NULLER](#), [PLACER](#), [st_char_get\(\)](#), [st_cmp_cs_eq\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [http_content_t::type](#), [http_data_t::value](#), [xml_dump_doc\(\)](#), [xml_encode\(\)](#), [xml_set_xpath_ns\(\)](#), [xml_set_xpath_property\(\)](#), and [http_page_t::xpath_ctx](#).

Referenced by [contact_business\(\)](#), and [contact_process\(\)](#).

5.446.1.2 void [contact_print_message](#) ([connection_t](#) *con, [chr_t](#) *branch, [chr_t](#) *message)

Display the contact or abuse notification form to the requesting user.

Note:

Both the contact and abuse forms operate on the underlying template found in "contact/message"

Parameters:

con a pointer to the connection object across which the response will be sent.

branch a null-terminated string specifying the type of contact, either "Contact" or "Abuse".

message a null-terminated string pointing to a custom message to be displayed to the user.

Returns:

This function returns no value.

Definition at line 18 of file contact.c.

References `con_write_st()`, `http_page_t::content`, `http_page_t::doc_obj`, `http_page_free()`, `http_page_get()`, `http_print_500()`, `http_print_500_log()`, `http_response_header()`, `NULLER`, `PLACER`, `st_cmp_cs_eq()`, `st_free()`, `st_length_get()`, `http_content_t::type`, `xml_dump_doc()`, `xml_set_xpath_ns()`, `xml_set_xpath_property()`, and `http_page_t::xpath_ctx`.

Referenced by `contact_abuse_checks()`, `contact_business()`, and `contact_print_form()`.

5.446.1.3 void contact_process (connection_t * con, chr_t * branch)

Process all user contact requests.

Parameters:

con a pointer to the connection object generating the contact request.

branch a null-terminated string specifying the request type (can be "Abuse" or "Contact").

Definition at line 122 of file contact.c.

References `con_secure()`, `contact_abuse_checks()`, `contact_business()`, `contact_print_form()`, `http_data_get()`, `HTTP_DATA_POST`, `HTTP_PARSE_PAIRS`, `http_print_301()`, `NULLER`, `PLACER`, and `st_cmp_cs_eq()`.

Referenced by `http_response()`.

5.447 src/web/contact/contact.h File Reference

Functions

- `bool_t contact_abuse_checks (connection_t *con, chr_t *branch)`
abuse.c
- `void contact_abuse_increment_history (connection_t *con)`
Increment the contact abuse history counter for an IP address.
- `void contact_business (connection_t *con, chr_t *branch)`
business.c
- `http_page_t * contact_business_add_error (chr_t *branch, uchr_t *xpath, uchr_t *id, uchr_t *message)`
Return the contact/abuse page with a marked error indicator for the user in the event of a user submission error.
- `bool_t contact_business_valid_email (stringer_t *email)`
Validate an email address.
- `void contact_print_form (connection_t *con, chr_t *branch, http_page_t *page)`
contact.c
- `void contact_print_message (connection_t *con, chr_t *branch, chr_t *message)`
Display the contact or abuse notification form to the requesting user.
- `void contact_process (connection_t *con, chr_t *branch)`
Process all user contact requests.

5.447.1 Function Documentation

5.447.1.1 `bool_t contact_abuse_checks (connection_t * con, chr_t * branch)`

abuse.c abuse.c

Note:

Each IP address will be limited to at most 2 contact requests in any 24-hour period.

Parameters:

con a pointer to the connection object of the remote host making the contact request.

branch a null-terminated string specifying where the contact request was directed ("Abuse" or "Contact").

Returns:

true if the specified connection failed the abuse check or false if it did not.

Definition at line 34 of file abuse.c.

References `cache_get_u64()`, `con_addr_presentation()`, `contact_print_message()`, `MANAGEDBUF`, `st_char_get()`, `st_length_int()`, and `st_sprint()`.

Referenced by `contact_process()`.

5.447.1.2 void contact_abuse_increment_history (connection_t * con)

Increment the contact abuse history counter for an IP address.

Parameters:

con a pointer to the connection object of the remote host making the contact request.

Returns:

This function returns no value.

Definition at line 15 of file abuse.c.

References cache_increment(), con_addr_presentation(), MANAGEDBUF, st_char_get(), st_length_int(), and st_sprint().

Referenced by contact_business().

5.447.1.3 void contact_business (connection_t * con, chr_t * branch)

business.c business.c

Parameters:

con the connection of the web client making the request.

branch a null-terminated string containing the destination of the contact form: either "Abuse" or "Contact".

Returns:

This function returns no value.

Definition at line 127 of file business.c.

References magma_t::abuse, magma_t::admin, con_addr_presentation(), magma_t::contact, contact_abuse_increment_history(), contact_business_add_error(), contact_business_valid_email(), contact_print_form(), contact_print_message(), http_data_get(), HTTP_DATA_POST, log_error, magma, MANAGEDBUF, NULLER, PLACER, smtp_send_message(), st_cmp_cs_eq(), st_free(), st_merge, st_replace(), and http_data_t::value.

Referenced by contact_process().

5.447.1.4 http_page_t* contact_business_add_error (chr_t * branch, uchr_t * xpath, uchr_t * id, uchr_t * message)

Return the contact/abuse page with a marked error indicator for the user in the event of a user submission error.

Parameters:

branch a null-terminated string containing the source of the error: "Abuse" or "Contact"

xpath a null-terminated string containing the xpath of the element in the returned page that should be colored red.

id a null-terminated string containing the id of the error text element to be added to the document.

message a null-terminated string containing the actual error message text to be displayed to the user.

Returns:

NULL on failure, or a pointer to the processed contact page with error message on success.

Definition at line 18 of file business.c.

References http_page_get(), NULLER, PLACER, st_cmp_cs_eq(), xml_node_add_sibling(), xml_node_free(), xml_node_new(), xml_node_set_content(), xml_node_set_property(), xml_xpath_eval(), and http_page_t::xpath_ctx.

Referenced by contact_business().

5.447.1.5 bool_t contact_business_valid_email (stringer_t * email)

Validate an email address.

Parameters:

email a managed string containing the email address to be validated.

Returns:

true if the specified email was valid and false if it was not.

Definition at line 60 of file business.c.

References length, lower_st(), st_char_get(), and st_length_get().

Referenced by config_validate_settings(), and contact_business().

5.447.1.6 void contact_print_form (connection_t * con, chr_t * branch, http_page_t * page)

[contact.c](#) [contact.c](#)

Parameters:

con the connection across which the form will be returned.

branch a null-terminated string specifying the type of contact, either "Contact" or "Abuse".

page a pointer to an http page containing the contact document to be populated with the user's name, email, and message.

Returns:

This function returns no value.

Definition at line 71 of file contact.c.

References con_write_st(), contact_print_message(), http_page_t::content, data, http_page_t::doc_obj, http_data_get(), HTTP_DATA_POST, http_page_free(), http_page_get(), http_response_header(), mm_cleanup(), NULLER, PLACER, st_char_get(), st_cmp_cs_eq(), st_free(), st_length_get(), http_content_t::type, http_data_t::value, xml_dump_doc(), xml_encode(), xml_set_xpath_ns(), xml_set_xpath_property(), and http_page_t::xpath_ctx.

Referenced by contact_business(), and contact_process().

5.447.1.7 void contact_print_message (connection_t * con, chr_t * branch, chr_t * message)

Display the contact or abuse notification form to the requesting user.

Note:

Both the contact and abuse forms operate on the underlying template found in "contact/message"

Parameters:

con a pointer to the connection object across which the response will be sent.

branch a null-terminated string specifying the type of contact, either "Contact" or "Abuse".

message a null-terminated string pointing to a custom message to be displayed to the user.

Returns:

This function returns no value.

Definition at line 18 of file contact.c.

References `con_write_st()`, `http_page_t::content`, `http_page_t::doc_obj`, `http_page_free()`, `http_page_get()`, `http_print_500()`, `http_print_500_log()`, `http_response_header()`, `NULLER`, `PLACER`, `st_cmp_cs_eq()`, `st_free()`, `st_length_get()`, `http_content_t::type`, `xml_dump_doc()`, `xml_set_xpath_ns()`, `xml_set_xpath_property()`, and `http_page_t::xpath_ctx`.

Referenced by `contact_abuse_checks()`, `contact_business()`, and `contact_print_form()`.

5.447.1.8 void contact_process (connection_t * *con*, chr_t * *branch*)

Process all user contact requests.

Parameters:

con a pointer to the connection object generating the contact request.

branch a null-terminated string specifying the request type (can be "Abuse" or "Contact").

Definition at line 122 of file contact.c.

References `con_secure()`, `contact_abuse_checks()`, `contact_business()`, `contact_print_form()`, `http_data_get()`, `HTTP_DATA_POST`, `HTTP_PARSE_PAIRS`, `http_print_301()`, `NULLER`, `PLACER`, and `st_cmp_cs_eq()`.

Referenced by `http_response()`.

5.448 src/web/json_api/endpoints.c File Reference

```
#include "magma.h"
```

Functions

- void [api_endpoint_auth](#) ([connection_t](#) *con)
- void [api_endpoint_register](#) ([connection_t](#) *con)
- void [api_endpoint_delete_user](#) ([connection_t](#) *con)
- void [api_endpoint_change_password](#) ([connection_t](#) *con)

5.448.1 Function Documentation

5.448.1.1 void [api_endpoint_auth](#) ([connection_t](#) * *con*)

Definition at line 42 of file endpoints.c.

References [api_error\(\)](#), [api_response\(\)](#), [auth_free\(\)](#), [auth_login\(\)](#), [cache_increment\(\)](#), [con_addr_presentation\(\)](#), [con_addr_subnet\(\)](#), [HTTP_COOKIE_SET](#), [HTTP_ERROR_400](#), [HTTP_ERROR_500](#), [HTTP_OK](#), [JSON_RPC_2_ERROR_SERVER_INTERNAL](#), [JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS](#), [json_unpack_ex_d](#), [auth_t::keys](#), [lock_get\(\)](#), [lock_release\(\)](#), [log_info](#), [log_pedantic](#), [MANAGED_BUF](#), [auth_t::master](#), [meta_get\(\)](#), [META_GET_NONE](#), [META_PROTOCOL_JSON](#), [meta_user_t](#), [NULLER](#), [PORTAL_ENDPOINT_ERROR_AUTH](#), [auth_t::salt](#), [auth_t::seasoning](#), [SESSION_STATE_AUTHENTICATED](#), [st_char_get\(\)](#), [st_length_int\(\)](#), [st_populated](#), [st_quick\(\)](#), [time_datestamp\(\)](#), [auth_t::tokens](#), [auth_t::username](#), [auth_t::usenum](#), and [auth_t::verification](#).

5.448.1.2 void [api_endpoint_change_password](#) ([connection_t](#) * *con*)

TODO: - wire up here

Definition at line 259 of file endpoints.c.

References [api_error\(\)](#), [HTTP_ERROR_400](#), [JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS](#), [json_unpack_ex_d](#), [log_pedantic](#), and [st_char_get\(\)](#).

5.448.1.3 void [api_endpoint_delete_user](#) ([connection_t](#) * *con*)

Definition at line 220 of file endpoints.c.

References [api_error\(\)](#), [api_response\(\)](#), [HTTP_ERROR_400](#), [HTTP_ERROR_422](#), [HTTP_ERROR_500](#), [HTTP_OK](#), [JSON_RPC_2_ERROR_SERVER_INTERNAL](#), [JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS](#), [json_unpack_ex_d](#), [log_pedantic](#), [mm_wipe\(\)](#), [ns_length_get\(\)](#), [st_char_get\(\)](#), and [stmt_exec_affected\(\)](#).

5.448.1.4 void [api_endpoint_register](#) ([connection_t](#) * *con*)

Definition at line 141 of file endpoints.c.

References [api_error\(\)](#), [api_response\(\)](#), [auth_sanitize_username\(\)](#), [HTTP_ERROR_400](#), [HTTP_ERROR_500](#), [HTTP_OK](#), [JSON_RPC_2_ERROR_SERVER_INTERNAL](#), [JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS](#), [json_unpack_ex_d](#), [log_pedantic](#), [ns_length_get\(\)](#), [NULLER](#), [pl_init\(\)](#), [PLACER](#), [placer_t](#), [register_abuse_increment_history\(\)](#), [register_data_insert_user\(\)](#), [st_cleanup](#), [st_cmp_ci_eq\(\)](#), [tran_commit\(\)](#), [tran_rollback\(\)](#), and [tran_start\(\)](#).

5.449 src/web/json_api/helpers.c File Reference

```
#include "magma.h"
```

Functions

- unsigned long [jansson_flags](#) (void)
- void [api_error](#) ([connection_t](#) *con, [int_t](#) http_code, [int_t](#) error_code, [chr_t](#) *message)
Return a json-rpc error response to the remote client.
- void [api_response](#) ([connection_t](#) *con, [int_t](#) http_code, [chr_t](#) *format,...)
Generate a json-rpc 2.0 response to a portal request.

5.449.1 Function Documentation

5.449.1.1 void [api_error](#) ([connection_t](#) *con, [int_t](#) http_code, [int_t](#) error_code, [chr_t](#) *message)

Return a json-rpc error response to the remote client.

Parameters:

- con** a pointer to the connection object across which the response will be sent.
- http_code** the http response code to be sent to the remote client in the response header.
- error_code** the numerical error code to be encoded in the json-rpc error message.
- message** a descriptive error string to be encoded in the json-rpc error message.

Returns:

This function returns no value.

Definition at line 25 of file helpers.c.

References [api_response](#)().

Referenced by [api_endpoint_auth](#)(), [api_endpoint_change_password](#)(), [api_endpoint_delete_user](#)(), and [api_endpoint_register](#)().

5.449.1.2 void [api_response](#) ([connection_t](#) *con, [int_t](#) http_code, [chr_t](#) *format, ...)

Generate a json-rpc 2.0 response to a portal request.

See also:

[json_vpack_ex](#)()

Note:

This function indents the json response if specified in the configuration, and also automatically decreases the reference count of any json object that was packed for the reply.

Parameters:

- con** a pointer to the connection object across which the portal response will be sent.
- format** a pointer to a format string specifying the construction of the json-rpc response.
- va_list** a variable arguments style list of parameters to be passed to the json packing function.

Returns:

This function returns no value.

Definition at line 40 of file helpers.c.

References `con_write_st()`, `HTTP_ERROR_500`, `http_response_header()`, `jansson_flags()`, `json_decref_d`, `json_dumps_d`, `json_vpack_ex_d`, `log_pedantic`, `magma`, `ns_append()`, `ns_free()`, `ns_length_get()`, `NULLER`, `PLACER`, `magma_t::portal`, and `magma_t::web`.

Referenced by `api_endpoint_auth()`, `api_endpoint_delete_user()`, `api_endpoint_register()`, and `api_error()`.

5.449.1.3 unsigned long jansson_flags (void)

Definition at line 10 of file helpers.c.

References `magma`, `magma_t::portal`, and `magma_t::web`.

Referenced by `api_response()`.

5.450 src/web/json_api/json_api.c File Reference

```
#include "magma.h"
```

Data Structures

- struct [api_lookup_t](#)

Functions

- void [json_api_dispatch](#) ([connection_t](#) *con)

The entry point for json API requests.

5.450.1 Function Documentation

5.450.1.1 void json_api_dispatch (connection_t * con)

The entry point for json API requests.

Parameters:

con The connection object corresponding to the web client making the request.

Returns:

This function returns no value.

TODO: - "merged" does not convey much useful information here

Definition at line 50 of file json_api.c.

References [api_lookup_t::callback](#), [con_localhost\(\)](#), [con_secure\(\)](#), [HTTP_MERGED](#), [http_parse_context\(\)](#), [HTTP_PORTAL](#), [json_integer_value_d](#), [json_loads_d](#), [json_object_get_d](#), [json_string_value_d](#), [log_error](#), [log_pedantic](#), [magma](#), [PLACER](#), [magma_t::portal](#), [sess_create\(\)](#), [st_char_get\(\)](#), [api_lookup_t::string](#), [uint64_conv_ns\(\)](#), and [magma_t::web](#).

Referenced by [http_response\(\)](#).

5.451 src/web/json_api/json_api.h File Reference

Functions

- void `json_api_dispatch` (`connection_t` *con)
The entry point for json API requests.
- void `api_endpoint_auth` (`connection_t` *con)
- void `api_endpoint_register` (`connection_t` *con)
- void `api_endpoint_delete_user` (`connection_t` *con)
- void `api_endpoint_change_password` (`connection_t` *con)
- void `api_error` (`connection_t` *con, `int_t` http_code, `int_t` error_code, `chr_t` *message)
Return a json-rpc error response to the remote client.
- void `api_response` (`connection_t` *con, `int_t` http_code, `chr_t` *format,...)
Generate a json-rpc 2.0 response to a portal request.

5.451.1 Function Documentation

5.451.1.1 void api_endpoint_auth (connection_t * con)

Definition at line 42 of file endpoints.c.

References `api_error()`, `api_response()`, `auth_free()`, `auth_login()`, `cache_increment()`, `con_addr_presentation()`, `con_addr_subnet()`, `HTTP_COOKIE_SET`, `HTTP_ERROR_400`, `HTTP_ERROR_500`, `HTTP_OK`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `auth_t::keys`, `lock_get()`, `lock_release()`, `log_info`, `log_pedantic`, `MANAGED_BUF`, `auth_t::master`, `meta_get()`, `META_GET_NONE`, `META_PROTOCOL_JSON`, `meta_user_t`, `NULLER`, `PORTAL_ENDPOINT_ERROR_AUTH`, `auth_t::salt`, `auth_t::seasoning`, `SESSION_STATE_AUTHENTICATED`, `st_char_get()`, `st_length_int()`, `st_populated`, `st_quick()`, `time_datestamp()`, `auth_t::tokens`, `auth_t::username`, `auth_t::usernum`, and `auth_t::verification`.

5.451.1.2 void api_endpoint_change_password (connection_t * con)

TODO: - wire up here

Definition at line 259 of file endpoints.c.

References `api_error()`, `HTTP_ERROR_400`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, and `st_char_get()`.

5.451.1.3 void api_endpoint_delete_user (connection_t * con)

Definition at line 220 of file endpoints.c.

References `api_error()`, `api_response()`, `HTTP_ERROR_400`, `HTTP_ERROR_422`, `HTTP_ERROR_500`, `HTTP_OK`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `mm_wipe()`, `ns_length_get()`, `st_char_get()`, and `stmt_exec_affected()`.

5.451.1.4 void api_endpoint_register (connection_t * con)

Definition at line 141 of file endpoints.c.

References `api_error()`, `api_response()`, `auth_sanitize_username()`, `HTTP_ERROR_400`, `HTTP_ERROR_500`, `HTTP_OK`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `ns_length_get()`, `NULLER`, `pl_init()`, `PLACER`, `placer_t`, `register_abuse_increment_history()`, `register_data_insert_user()`, `st_cleanup`, `st_cmp_ci_eq()`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

5.451.1.5 void api_error (connection_t * con, int_t http_code, int_t error_code, chr_t * message)

Return a json-rpc error response to the remote client.

Parameters:

- con* a pointer to the connection object across which the response will be sent.
- http_code* the http response code to be sent to the remote client in the response header.
- error_code* the numerical error code to be encoded in the json-rpc error message.
- message* a descriptive error string to be encoded in the json-rpc error message.

Returns:

This function returns no value.

Definition at line 25 of file helpers.c.

References `api_response()`.

Referenced by `api_endpoint_auth()`, `api_endpoint_change_password()`, `api_endpoint_delete_user()`, and `api_endpoint_register()`.

5.451.1.6 void api_response (connection_t * con, int_t http_code, chr_t * format, ...)

Generate a json-rpc 2.0 response to a portal request.

See also:

`json_vpack_ex()`

Note:

This function indents the json response if specified in the configuration, and also automatically decreases the reference count of any json object that was packed for the reply.

Parameters:

- con* a pointer to the connection object across which the portal response will be sent.
- format* a pointer to a format string specifying the construction of the json-rpc response.
- va_list* a variable arguments style list of parameters to be passed to the json packing function.

Returns:

This function returns no value.

Definition at line 40 of file helpers.c.

References `con_write_st()`, `HTTP_ERROR_500`, `http_response_header()`, `jansson_flags()`, `json_decref_d`, `json_dumps_d`, `json_vpack_ex_d`, `log_pedantic`, `magma`, `ns_append()`, `ns_free()`, `ns_length_get()`, `NULLER`, `PLACER`, `magma_t::portal`, and `magma_t::web`.

Referenced by `api_endpoint_auth()`, `api_endpoint_delete_user()`, `api_endpoint_register()`, and `api_error()`.

5.451.1.7 void json_api_dispatch (connection_t * con)

The entry point for json API requests.

Parameters:

- con* The connection object corresponding to the web client making the request.

Returns:

This function returns no value.

TODO: - "merged" does not convey much useful information here

Definition at line 50 of file json_api.c.

References `api_lookup_t::callback`, `con_localhost()`, `con_secure()`, `HTTP_MERGED`, `http_parse_context()`, `HTTP_PORTAL`, `json_integer_value_d`, `json_loads_d`, `json_object_get_d`, `json_string_value_d`, `log_error`, `log_pedantic`, `magma`, `PLACER`, `magma_t::portal`, `sess_create()`, `st_char_get()`, `api_lookup_t::string`, `uint64_conv_ns()`, and `magma_t::web`.

Referenced by `http_response()`.

5.452 src/web/portal/endpoint.c File Reference

```
#include "magma.h"
#include "methods.h"
```

Functions

- void [portal_endpoint_sort](#) (void)
Sort the web portal JSON command table to be ready for binary searches.
- [int_t portal_endpoint_compare](#) (const void *compare, const void *command)
Internal bsearch() comparison function for dispatching the proper portal handler.
- [bool_t portal_validate_request](#) ([connection_t](#) *con, int err_mask, [chr_t](#) *method, [bool_t](#) has_params, size_t nparams)
Verify that a session underlying a portal request has been authenticated.
- void [portal_endpoint_error](#) ([connection_t](#) *con, [int_t](#) http_code, [int_t](#) error_code, [chr_t](#) *message)
Return a json-rpc error response to the remote client.
- void [portal_endpoint_response](#) ([connection_t](#) *con, [chr_t](#) *format,...)
Generate a json-rpc 2.0 response to a portal request.
- void [portal_endpoint_auth](#) ([connection_t](#) *con)
Obtain credentials and log a user into portal session in response to an "auth" json-rpc portal request.
- void [portal_endpoint_logout](#) ([connection_t](#) *con)
Log a user out of a portal session in response to a "logout" json-rpc portal request.
- void [portal_endpoint_alert_list](#) ([connection_t](#) *con)
Get the list of a user's alert messages in response to an "alert.list" json-rpc portal request.
- void [portal_endpoint_alert_acknowledge](#) ([connection_t](#) *con)
Process user alert message acknowledgements in response to an "alert.acknowledge" json-rpc portal request.
- void [portal_endpoint_aliases](#) ([connection_t](#) *con)
Return the list of a user account's aliases in response to an "aliases" json-rpc portal request.
- void [portal_endpoint_cookies](#) ([connection_t](#) *con)
Return whether or not a user is using cookies, in response to a "cookies" json-rpc portal request.
- void [portal_endpoint_folders_add](#) ([connection_t](#) *con)
Create a new folder in response to a "folders.add" json-rpc portal request.
- void [portal_endpoint_folders_list](#) ([connection_t](#) *con)
Return a list of a user's folders in response to a "folders.list" json-rpc portal request.
- void [portal_endpoint_folders_remove](#) ([connection_t](#) *con)
Remove a user's folder in response to a json-rpc "folders.remove" portal request.
- void [portal_endpoint_folders_rename](#) ([connection_t](#) *con)
Rename a user's folder in response to a json-rpc "folders.rename" portal request.

- void [portal_endpoint_folders_tags](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_list](#) ([connection_t](#) *con)

Retrieve a list of the user's messages in response to a json-rpc "messages.list" portal request.

- void [portal_endpoint_messages_copy](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_flag](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_move](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_remove](#) ([connection_t](#) *con)

Remove a user's mail message in response to a "messages.remove" json-rpc portal request.

- void [portal_endpoint_messages_tags](#) ([connection_t](#) *con)

Get all of the message tags used by a specified user in response to a "messages.tags" json-rpc portal request.

- void [portal_endpoint_messages_tag](#) ([connection_t](#) *con)

Perform a tag operation on a user's message, in response to a "messages.tag" json-rpc portal request.

- void [portal_endpoint_messages_load](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_compose](#) ([connection_t](#) *con)

Compose a new message in response to a "messages.compose" json-rpc portal request.

- void [portal_endpoint_attachments_add](#) ([connection_t](#) *con)

Add an attachment to a message being composed in response to an "attachments.add" json-rpc portal request.

- void [portal_endpoint_attachments_remove](#) ([connection_t](#) *con)

Remove an attachment from a message being composed in response to an "attachments.remove" json-rpc portal request.

- void [portal_endpoint_messages_send](#) ([connection_t](#) *con)

Send a composed message in response to a "messages.send" json-rpc portal request.

- void [portal_endpoint_contacts_copy](#) ([connection_t](#) *con)
- void [portal_endpoint_contacts_move](#) ([connection_t](#) *con)

Move a user's contact entry to another contacts folder in response to a "contacts.move" json-rpc portal request.

- void [portal_endpoint_contacts_remove](#) ([connection_t](#) *con)

Remove a user's contact entry in response to a "contacts.remove" json-rpc portal request.

- void [portal_endpoint_contacts_list](#) ([connection_t](#) *con)

List a user's contacts in response to a "contacts.list" json-rpc portal request.

- void [portal_endpoint_contacts_add](#) ([connection_t](#) *con)

Add a contact in response to a "contacts.add" json-rpc portal request.

- void [portal_endpoint_contacts_edit](#) ([connection_t](#) *con)
- void [portal_endpoint_config_edit](#) ([connection_t](#) *con)

Retrieve all of a user's config options and return them to the remote client in response to a portal "contacts.load" json-rpc request.

- void [portal_endpoint_config_load](#) ([connection_t](#) *con)

Retrieve all of a user's config options and return them to the remote client in response to a portal "contacts.load" json-rpc request.

- void [portal_endpoint_contacts_load](#) ([connection_t](#) *con)

Return information for a portal "contacts.load" json-rpc request.

- void [portal_endpoint_ad](#) ([connection_t](#) *con)

Return advertising information for a portal "ad" json-rpc request.

- void [portal_endpoint_search](#) ([connection_t](#) *con)
- void [portal_endpoint_scrape_add](#) ([connection_t](#) *con)
- void [portal_endpoint_scrape](#) ([connection_t](#) *con)
- void [portal_endpoint_attachments_progress](#) ([connection_t](#) *con)
- void [portal_settings_identity](#) ([connection_t](#) *con)

Return information for a portal "settings.identity" json-rpc request.

- void [portal_meta](#) ([connection_t](#) *con)

Return information for a portal "meta" json-rpc request.

- void [portal_settings_changepass](#) ([connection_t](#) *con)

Change a user's password in response to a portal "settings.changepass" json-rpc request.

- void [portal_endpoint](#) ([connection_t](#) *con)

The entry point for camel requests sent to the portal.

- [attachment_t](#) * [portal_get_upload_attachment](#) ([connection_t](#) *con)

Get a user's attachment to a message composition uploaded via multipart form data.

- void [portal_upload](#) ([connection_t](#) *con)

Process uploaded attachments for messages composed in conjunction with the portal interface.

- void [portal_debug](#) ([connection_t](#) *con)

A portal debug function that will be disabled and/or deleted completely in production.

5.452.1 Function Documentation

5.452.1.1 void [portal_debug](#) ([connection_t](#) * con)

A portal debug function that will be disabled and/or deleted completely in production.

Note:

The debug output is displayed LOCALLY in the Magma console.

Definition at line 6254 of file endpoint.c.

5.452.1.2 void [portal_endpoint](#) ([connection_t](#) * con)

The entry point for camel requests sent to the portal. [endpoint.c](#)

Parameters:

con the connection object corresponding to the web client making the request.

Returns:

This function returns no value.

Definition at line 5913 of file endpoint.c.

References `command`, `command_t`, `con_localhost()`, `con_secure()`, `connection_t`, `HTTP_ERROR_500`, `HTTP_MERGED`, `http_parse_context()`, `HTTP_PORTAL`, `http_print_301()`, `http_response_header()`, `json_integer_value_d`, `json_loads_d`, `json_object_get_d`, `JSON_RPC_2_ERROR_PARSE_MALFORMED`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL`, `JSON_RPC_2_ERROR_SERVER_REQUEST`, `json_string_value_d`, `json_type_string_d`, `log_pedantic`, `magma`, `ns_length_get()`, `PLACER`, `magma_t::portal`, `portal_endpoint_compare()`, `portal_endpoint_error()`, `portal_methods`, `sess_create()`, `st_char_get()`, `uint64_conv_ns()`, and `magma_t::web`.

Referenced by `http_response()`.

5.452.1.3 `void portal_endpoint_ad (connection_t * con)`

Return advertising information for a portal "ad" json-rpc request. ^ finished / work area v ^ work area / stubs v

Note:

This function is not implemented and may be removed entirely in the future.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2888 of file endpoint.c.

References `json_object_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `log_pedantic`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_AD`, `portal_endpoint_response()`, and `portal_validate_request()`.

5.452.1.4 `void portal_endpoint_alert_acknowledge (connection_t * con)`

Process user alert message acknowledgements in response to an "alert.acknowledge" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 444 of file endpoint.c.

References `count`, `json_array_get_d`, `json_array_size_d`, `json_integer_value_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `meta_data_acknowledge_alert()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_ALERT_ACKNOWLEDGE`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, `st_length_get()`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

5.452.1.5 `void portal_endpoint_alert_list (connection_t * con)`

Get the list of a user's alert messages in response to an "alert.list" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 396 of file endpoint.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `log_pedantic`, `meta_alert_t`, `meta_data_fetch_alerts()`, `portal_endpoint_error()`, `POTAL_ENDPOINT_ERROR_ALERT_LIST`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_char_get()`.

5.452.1.6 void portal_endpoint_aliases (connection_t * con)

Return the list of a user account's aliases in response to an "aliases" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 514 of file endpoint.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `json_true_d`, `log_pedantic`, `meta_alias_t`, `portal_endpoint_error()`, `POTAL_ENDPOINT_ERROR_ALIASES`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_char_get()`.

5.452.1.7 void portal_endpoint_attachments_add (connection_t * con)

Add an attachment to a message being composed in response to an "attachments.add" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2002 of file endpoint.c.

References `attachment_t`, `composition_t`, `inx_find()`, `inx_insert()`, `json_object_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `mm_alloc()`, `mutex_lock()`, `mutex_unlock()`, `ns_length_get()`, `portal_endpoint_error()`, `POTAL_ENDPOINT_ERROR_ATTACHMENTS_ADD`, `POTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `sess_release_attachment()`, `st_char_get()`, `st_import()`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

5.452.1.8 void portal_endpoint_attachments_progress (connection_t * con)

Definition at line 5778 of file endpoint.c.

References `con_write_st()`, `HTTP_ERROR_500`, `http_response_header()`, `PLACER`, `POTAL_ENDPOINT_ERROR_ATTACHMENTS_PROGRESS`, `st_aprint()`, `st_free()`, and `st_length_get()`.

5.452.1.9 void portal_endpoint_attachments_remove (connection_t * con)

Remove an attachment from a message being composed in response to an "attachments.remove" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2080 of file endpoint.c.

References attachment_t, composition_t, inx_delete(), inx_find(), JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_error, log_pedantic, M_TYPE_UINT64, mutex_lock(), mutex_unlock(), portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ATTACHMENTS_REMOVE, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.452.1.10 void portal_endpoint_auth (connection_t * con)

Obtain credentials and log a user into portal session in response to an "auth" json-rpc portal request.

Parameters:

con a pointer to the connection object of the user attempting to log in.

Returns:

This function returns no value.

Definition at line 194 of file endpoint.c.

References auth_free(), AUTH_LOCK_ABUSE, AUTH_LOCK_ADMIN, AUTH_LOCK_EXPIRED, AUTH_LOCK_INACTIVITY, AUTH_LOCK_USER, auth_login(), cache_increment(), con_addr_presentation(), con_addr_subnet(), con_localhost(), con_secure(), HTTP_COOKIE_SET, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, auth_t::keys, lock_get(), lock_release(), auth_t::locked, log_info, log_pedantic, MANAGEDBUF, auth_t::master, meta_get(), META_GET_ALIASES, META_GET_CONTACTS, META_GET_FOLDERS, META_GET_KEYS, META_GET_MESSAGES, meta_inx_remove(), META_PROTOCOL_WEB, META_USER_ENCRYPT_DATA, meta_user_t, META_USER_TLS, NULLER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_AUTH, PORTAL_ENDPOINT_ERROR_MODE, portal_endpoint_response(), auth_t::salt, auth_t::seasoning, SESSION_STATE_AUTHENTICATED, SESSION_STATE_NEUTRAL, st_char_get(), st_length_int(), st_populated, st_quick(), auth_t::status, time_datestamp(), auth_t::tokens, auth_t::username, auth_t::usernum, and auth_t::verification.

5.452.1.11 int_t portal_endpoint_compare (const void * compare, const void * command)

Internal bsearch() comparison function for dispatching the proper portal handler.

Parameters:

compare a pointer to a command_t object with a portal method name for string comparison.

command a pointer to a command_t object with a portal method name for string comparison.

Returns:

Definition at line 36 of file endpoint.c.

References command_t, PLACER, and st_cmp_ci_eq().

Referenced by portal_endpoint(), and portal_endpoint_sort().

5.452.1.12 void portal_endpoint_config_edit (connection_t * con)

Retrieve all of a user's config options and return them to the remote client in response to a portal "contacts.load" json-rpc request.

Note:

Requires a user to be authenticated; failure will be communicated to the client via a json response.

Parameters:

con a pointer to a connection object over which the json response will be sent.

Returns:

This function returns no value.

HIGH: We aren't checking/validating the config entries! And we should probably do our own duplication checks to avoid placing unnecessary load on the database.

Definition at line 2739 of file endpoint.c.

References content, json_object_iter_d, json_object_iter_key_d, json_object_iter_next_d, json_object_iter_value_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, json_string_value_d, NULLER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_CONFIG_EDIT, portal_endpoint_response(), portal_validate_request(), status, user_config_create(), user_config_edit(), user_config_free(), and user_config_t.

5.452.1.13 void portal_endpoint_config_load (connection_t * con)

Retrieve all of a user's config options and return them to the remote client in response to a portal "contacts.load" json-rpc request.

Note:

Requires a user to be authenticated; failure will be communicated to the client via a json response.

Parameters:

con a pointer to a connection object over which the json response will be sent.

Returns:

This function returns no value.

Definition at line 2798 of file endpoint.c.

References JSON_RPC_2_ERROR_SERVER_INTERNAL, portal_config_collection(), portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_CONFIG_LOAD, portal_endpoint_response(), portal_validate_request(), user_config_create(), user_config_free(), and user_config_t.

5.452.1.14 void portal_endpoint_contacts_add (connection_t * con)

Add a contact in response to a "contacts.add" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

HIGH: We aren't checking/validating the contact details! And we should probably do our own duplication checks to avoid placing unnecessary load on the database.

Definition at line 2546 of file endpoint.c.

References `contact_create()`, `contact_edit()`, `contact_free()`, `contact_t`, `contact_validate_detail()`, `contact_validate_name()`, `content`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_insert()`, `json_object_get_d`, `json_object_iter_d`, `json_object_iter_key_d`, `json_object_iter_next_d`, `json_object_iter_value_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_string_value_d`, `json_type_string_d`, `json_unpack_ex_d`, `log_pedantic`, `M_TYPE_UINT64`, `magma_folder_find_number()`, `meta_user_rlock()`, `meta_user_unlock()`, `meta_user_wlock()`, `NULLER`, `OBJECT_CONTACTS`, `PLACER`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION`, `PORTAL_ENDPOINT_ERROR_CONTACTS_ADD`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `sess_serial_check()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_cmp_cs_eq()`, `st_length_get()`, `status`, `multi_t::u64`, and `multi_t::val`.

5.452.1.15 `void portal_endpoint_contacts_copy (connection_t * con)`

Note:

If the source and target folder are equal, a copy will be made with a different name.

Definition at line 2252 of file endpoint.c.

References `contact_create()`, `contact_detail_t`, `contact_edit()`, `contact_find_number()`, `contact_free()`, `contact_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_insert()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `magma_folder_find_number()`, `meta_user_rlock()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `PLACER`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION`, `PORTAL_ENDPOINT_ERROR_CONTACTS_COPY`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `sess_serial_check()`, `st_char_get()`, `st_cleanup`, `st_length_get()`, `st_merge`, `multi_t::u64`, and `multi_t::val`.

5.452.1.16 `void portal_endpoint_contacts_edit (connection_t * con)`

HIGH: We aren't checking/validating the contact details! And we should probably do our own duplication checks to avoid placing unnecessary load on the database.

Definition at line 2662 of file endpoint.c.

References `contact_edit()`, `contact_find_number()`, `contact_t`, `content`, `json_object_iter_d`, `json_object_iter_key_d`, `json_object_iter_next_d`, `json_object_iter_value_d`, `json_object_size_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_string_value_d`, `json_unpack_ex_d`, `log_pedantic`, `magma_folder_find_number()`, `meta_user_rlock()`, `meta_user_unlock()`, `meta_user_wlock()`, `NULLER`, `OBJECT_CONTACTS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_EDIT`, `PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `sess_serial_check()`, `st_char_get()`, `st_length_get()`, and `status`.

5.452.1.17 `void portal_endpoint_contacts_list (connection_t * con)`

List a user's contacts in response to a "contacts.list" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2477 of file endpoint.c.

References `contact_detail_t`, `contact_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_object_set_new_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_string_d`, `json_unpack_ex_d`, `log_pedantic`, `magma_folder_find_number()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_LIST`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, and `st_length_get()`.

5.452.1.18 void portal_endpoint_contacts_load (connection_t * con)

Return information for a portal "contacts.load" json-rpc request.

Note:

This function returns the all the stored details about a specified contact entry.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2834 of file `endpoint.c`.

References `contact_t`, `inx_find()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `M_TYPE_UINT64`, `magma_folder_find_number()`, `portal_contact_details()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_LOAD`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

5.452.1.19 void portal_endpoint_contacts_move (connection_t * con)

Move a user's contact entry to another contacts folder in response to a "contacts.move" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2359 of file `endpoint.c`.

References `contact_find_number()`, `contact_move()`, `contact_t`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `magma_folder_find_number()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_MOVE`, `PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `sess_serial_check()`, `st_char_get()`, `st_length_get()`, and `status`.

5.452.1.20 void portal_endpoint_contacts_remove (connection_t * con)

Remove a user's contact entry in response to a "contacts.remove" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2422 of file endpoint.c.

References `contact_delete()`, `contact_find_number()`, `contact_t`, `inx_delete()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `M_TYPE_UINT64`, `magma_folder_find_number()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_REMOVE`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `sess_serial_check()`, `st_char_get()`, `st_length_get()`, `status`, `multi_t::u64`, and `multi_t::val`.

5.452.1.21 `void portal_endpoint_cookies (connection_t * con)`

Return whether or not a user is using cookies, in response to a "cookies" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 564 of file endpoint.c.

References `PLACER`, `PORTAL_ENDPOINT_ERROR_COOKIES`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_cmp_ci_starts()`.

5.452.1.22 `void portal_endpoint_error (connection_t * con, int_t http_code, int_t error_code, chr_t * message)`

Return a json-rpc error response to the remote client.

Parameters:

con a pointer to the connection object across which the response will be sent.

http_code the http response code to be sent to the remote client in the response header.

error_code the numerical error code to be encoded in the json-rpc error message.

message a descriptive error string to be encoded in the json-rpc error message.

Returns:

This function returns no value.

Definition at line 124 of file endpoint.c.

References `con_write_st()`, `HTTP_ERROR_500`, `http_response_header()`, `json_decref_d`, `json_dumps_d`, `json_pack_ex_d`, `log_options`, `log_pedantic`, `M_LOG_CRITICAL`, `M_LOG_STACK_TRACE`, `magma`, `ns_append()`, `ns_free()`, `ns_length_get()`, `NULLER`, `PLACER`, `magma_t::portal`, and `magma_t::web`.

Referenced by `portal_endpoint()`, `portal_endpoint_ad()`, `portal_endpoint_alert_acknowledge()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_auth()`, `portal_endpoint_config_edit()`, `portal_endpoint_config_load()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_folders_add()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_remove()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_endpoint_scrape()`, `portal_endpoint_scrape_add()`, `portal_folder_contacts_add()`, `portal_folder_contacts_remove()`, `portal_folder_mail_add()`, `portal_folder_mail_remove()`, `portal_meta()`, `portal_settings_changepass()`, `portal_settings_identity()`, and `portal_validate_request()`.

5.452.1.23 void portal_endpoint_folders_add (connection_t * con)

Create a new folder in response to a "folders.add" json-rpc portal request.

Note:

Currently only contexts for new mail folders and contacts folders are supported.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 587 of file endpoint.c.

References count, json_object_size_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, ns_length_get(), NULLER, PORTAL_ENDPOINT_CONTEXT_CONTACTS, PORTAL_ENDPOINT_CONTEXT_INVALID, PORTAL_ENDPOINT_CONTEXT_MAIL, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_FOLDERS_ADD, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, portal_folder_contacts_add(), portal_folder_mail_add(), portal_parse_context(), portal_validate_request(), st_char_get(), st_free(), st_import(), and st_length_get().

5.452.1.24 void portal_endpoint_folders_list (connection_t * con)

Return a list of a user's folders in response to a "folders.list" json-rpc portal request.

Note:

Currently only contexts for new mail folders and contacts folders are supported.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

LOW: User created folder names _could_ have a NULL byte. How should we handle that?

LOW: User created folder names _could_ have a NULL byte. How should we handle that?

Definition at line 649 of file endpoint.c.

References inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), json_array_append_new_d, json_array_d, json_decref_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, magma_folder_t, meta_folder_t, NULLER, PLACER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_FOLDERS_LIST, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, portal_endpoint_response(), portal_validate_request(), st_char_get(), st_cmp_ci_eq(), and st_length_get().

5.452.1.25 void portal_endpoint_folders_remove (connection_t * con)

Remove a user's folder in response to a json-rpc "folders.remove" portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 732 of file endpoint.c.

References JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, NULLER, PORTAL_ENDPOINT_CONTEXT_CONTACTS, PORTAL_ENDPOINT_CONTEXT_INVALID, PORTAL_ENDPOINT_CONTEXT_MAIL, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, portal_folder_contacts_remove(), portal_folder_mail_remove(), portal_parse_context(), portal_validate_request(), st_char_get(), and st_length_get().

5.452.1.26 void portal_endpoint_folders_rename (connection_t * con)

Rename a user's folder in response to a json-rpc "folders.rename" portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

HIGH: Use the context and act appropriately!

Definition at line 773 of file endpoint.c.

References contact_folder_rename(), imap_folder_rename(), JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, magma_folder_find_number(), magma_folder_name(), magma_folder_t, meta_folder_t, meta_folders_by_number(), meta_folders_name(), meta_user_unlock(), meta_user_wlock(), ns_length_get(), NULLER, OBJECT_CONTACTS, OBJECT_FOLDERS, PORTAL_ENDPOINT_CONTEXT_CONTACTS, PORTAL_ENDPOINT_CONTEXT_INVALID, PORTAL_ENDPOINT_CONTEXT_MAIL, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION, PORTAL_ENDPOINT_ERROR_FOLDERS_RENAME, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_parse_context(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_free(), st_import(), and st_length_get().

5.452.1.27 void portal_endpoint_folders_tags (connection_t * con)

HIGH: Right now we only have indexes with the mail folders so other context types generate an empty array result.

Definition at line 918 of file endpoint.c.

References meta_stats_tag_t::count, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_free(), inx_t, json_decref_d, json_integer_d, json_object_d, json_object_set_new_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, meta_folders_by_number(), meta_folders_stats_tags(), meta_user_rlock(), meta_user_unlock(), NULLER, PORTAL_ENDPOINT_CONTEXT_MAIL, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_FOLDERS_TAGS, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_parse_context(), portal_validate_request(), st_char_get(), st_length_get(), stats, and meta_stats_tag_t::tag.

5.452.1.28 void portal_endpoint_logout (connection_t * con)

Log a user out of a portal session in response to a "logout" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 374 of file endpoint.c.

References HTTP_COOKIE_DELETE, META_PROTOCOL_WEB, meta_user_ref_dec(), PORTAL_ENDPOINT_ERROR_LOGOUT, portal_endpoint_response(), portal_validate_request(), and SESSION_STATE_TERMINATED.

5.452.1.29 void portal_endpoint_messages_compose (connection_t * con)

Compose a new message in response to a "messages.compose" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 1942 of file endpoint.c.

References composition_t, inx_alloc(), inx_find(), inx_insert(), json_object_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, log_error, M_INX_LINKED, M_TYPE_UINT64, mm_alloc(), mutex_lock(), mutex_unlock(), portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_MESSAGES_COMPOSE, portal_endpoint_response(), portal_validate_request(), sess_release_attachment(), sess_release_composition(), multi_t::u64, and multi_t::val.

5.452.1.30 void portal_endpoint_messages_copy (connection_t * con)

HIGH: If a multiple message copy fails in the middle, go back and remove any messages that may have been copied before the error.

Definition at line 1114 of file endpoint.c.

References count, inx_find(), json_array_append_new_d, json_array_d, json_array_get_d, json_array_size_d, json_decref_d, json_integer_value_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_TYPE_UINT64, meta_folders_by_number(), META_LOCKED, meta_message_t, meta_messages_copier(), meta_messages_update_sequences(), meta_user_unlock(), meta_user_wlock(), OBJECT_MESSAGES, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_MESSAGES_COPY, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.452.1.31 void portal_endpoint_messages_flag (connection_t * con)

HIGH: This function needs restructuring pretty badly. We are setting collection to NULL so we can add a cleanup call below. But if the function works as intended the reference is stolen by the response. but if an error occurs and the we don't return a response the collection ends up as a memory leak without the cleanup call below.

TODO: Were holding onto the user lock while waiting on the response to be sent over the wire; and this function could probably use a rethink.

HIGH: See the sister comment a few lines back!

Definition at line 1203 of file endpoint.c.

References count, inx_alloc(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_find(), inx_free(), inx_insert(), inx_t, json_array_append_new_d, json_array_d, json_array_get_d, json_array_size_d, json_decref_d, json_integer_value_d, json_object_size_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_INX_LINKED, M_TYPE_UINT64, MAIL_STATUS_SYSTEM_FLAGS, MAIL_STATUS_USER_FLAGS, meta_data_flags_add(), meta_data_flags_remove(), meta_data_flags_replace(), meta_folders_by_number(), meta_message_t, meta_user_rlock(), meta_user_unlock(), meta_user_wlock(), NULLER, OBJECT_MESSAGES, PLACER, PORTAL_ENDPOINT_ACTION_ADD, PORTAL_ENDPOINT_ACTION_LIST, PORTAL_ENDPOINT_ACTION_REMOVE, PORTAL_ENDPOINT_ACTION_REPLACE, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, PORTAL_ENDPOINT_ERROR_MESSAGES_FLAG, PORTAL_ENDPOINT_ERROR_REFERENCE, PORTAL_ENDPOINT_ERROR_SYSTEM_FLAG, portal_endpoint_response(), portal_message_flags_array(), portal_parse_flags(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_cmp_ci_eq(), st_length_get(), multi_t::u64, and multi_t::val.

5.452.1.32 void portal_endpoint_messages_list (connection_t * con)

Retrieve a list of the user's messages in response to a json-rpc "messages.list" portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

LOW: Add the ability to track the recipient email address for a message, even if its not provided in the To field.

LOW: Add snippet support.

Definition at line 991 of file endpoint.c.

References ar_field_st(), ar_length_get(), count, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), json_array_append_new_d, json_array_d, json_decref_d, json_object_size_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_string_d, json_unpack_ex_d, log_pedantic, mail_header_fetch_cleaned(), mail_load_header(), meta_message_t, meta_user_rlock(), meta_user_unlock(), PLACER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_MESSAGES_LIST, portal_endpoint_response(), portal_message_flags_array(), portal_validate_request(), st_char_get(), st_cleanup, st_free(), st_import(), and st_length_get().

5.452.1.33 void portal_endpoint_messages_load (connection_t * con)

Definition at line 1836 of file endpoint.c.

References data, inx_find(), json_decref_d, json_object_d, json_object_set_new_d, json_object_size_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_TYPE_UINT64, mail_destroy(), mail_load_message(), mail_mime_update(), meta_message_t, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, PORTAL_ENDPOINT_ERROR_MESSAGES_LOAD, PORTAL_ENDPOINT_ERROR_READ, PORTAL_ENDPOINT_ERROR_REFERENCE, PORTAL_ENDPOINT_MESSAGE_ATTACHMENTS, PORTAL_ENDPOINT_MESSAGE_BODY, PORTAL_ENDPOINT_MESSAGE_HEADER, PORTAL_ENDPOINT_MESSAGE_INFO, PORTAL_ENDPOINT_MESSAGE_META, PORTAL_ENDPOINT_MESSAGE_SECURITY, PORTAL_ENDPOINT_MESSAGE_SERVER, PORTAL_ENDPOINT_MESSAGE_SOURCE, portal_endpoint_response(), portal_message_attachments(), portal_message_body(), portal_message_header(), portal_message_info(), portal_message_meta(), portal_message_security(), portal_message_server(), portal_message_source(), portal_parse_sections(), portal_validate_request(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.452.1.34 void portal_endpoint_messages_move (connection_t * con)

HIGH: If a multiple message copy fails in the middle, go back and remove any messages that may have been copied before the error.

Definition at line 1404 of file endpoint.c.

References count, inx_find(), json_array_get_d, json_array_size_d, json_integer_value_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_TYPE_UINT64, meta_folders_by_number(), META_LOCKED, meta_message_t, meta_messages_mover(), meta_messages_update_sequences(), meta_user_unlock(), meta_user_wlock(), OBJECT_MESSAGES, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, PORTAL_ENDPOINT_ERROR_MESSAGES_MOVE, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.452.1.35 void portal_endpoint_messages_remove (connection_t * con)

Remove a user's mail message in response to a "messages.remove" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 1500 of file endpoint.c.

References `count`, `inx_delete()`, `inx_find()`, `json_array_get_d`, `json_array_size_d`, `json_integer_value_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `M_TYPE_UINT64`, `mail_remove_message()`, `meta_folders_by_number()`, `meta_message_t`, `meta_messages_update_sequences()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_MESSAGES`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_MESSAGES_REMOVE`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `serial_get()`, `serial_increment()`, `sess_trigger()`, `st_char_get()`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

5.452.1.36 void portal_endpoint_messages_send (connection_t * con)

Send a composed message in response to a "messages.send" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2136 of file endpoint.c.

References `composition_t`, `inx_delete()`, `inx_find()`, `inx_free()`, `inx_t`, `json_object_size_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`, `mutex_lock()`, `mutex_unlock()`, `NULLER`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION`, `PORTAL_ENDPOINT_ERROR_MESSAGES_SEND`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_outbound_checks()`, `portal_parse_json_str_array()`, `portal_smtp_create_data()`, `portal_smtp_relay_message()`, `portal_validate_request()`, `st_char_get()`, `st_free()`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

5.452.1.37 void portal_endpoint_messages_tag (connection_t * con)

Perform a tag operation on a user's message, in response to a "messages.tag" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

LOW: We don't need to add the flag to every message. Just the ones that don't already have the flag.

LOW: Like the line, were not checking whether the message even needs to have the flag removed. Were also not handling remove requests that result in a message having no tags.

If this is going supposed to be a replace operation, we truncate all of the message tags so we can insert the replacements below.

TODO: This is ugly. Were rebuilding the tags array from the database on each iteration because its easier than trying to figure out which slot needs to be removed.

TODO: If the update is triggered before the tagged flag is added to the database the refresh might not pull the data for who recently got tagged! We need to need to look for this type of sequence bug elsewhere too.

LOW: Were holding onto the user lock while were waiting on the response to be sent over the wire; and this function could probably use a rethink.

Definition at line 1640 of file endpoint.c.

References `ar_field_st()`, `ar_free()`, `ar_length_get()`, `inx_alloc()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_find()`, `inx_free()`, `inx_insert()`, `inx_t`, `json_array_append_new_d`, `json_array_d`, `json_array_get_d`, `json_array_size_d`, `json_decref_d`, `json_integer_value_d`, `json_object_size_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_string_d`, `json_string_value_d`, `json_unpack_ex_d`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `MAIL_STATUS_TAGGED`, `meta_data_delete_tag()`, `meta_data_fetch_message_tags()`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_insert_tag()`, `meta_data_truncate_tags()`, `meta_folders_by_number()`, `meta_message_t`, `meta_user_rlock()`, `meta_user_serial_check()`, `meta_user_unlock()`, `meta_user_wlock()`, `NULLER`, `OBJECT_MESSAGES`, `PLACER`, `PORTAL_ENDPOINT_ACTION_ADD`, `PORTAL_ENDPOINT_ACTION_LIST`, `PORTAL_ENDPOINT_ACTION_REMOVE`, `PORTAL_ENDPOINT_ACTION_REPLACE`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION`, `PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD`, `PORTAL_ENDPOINT_ERROR_MESSAGES_TAG`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

5.452.1.38 void portal_endpoint_messages_tags (connection_t * con)

Get all of the message tags used by a specified user in response to a "messages.tags" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 1589 of file endpoint.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `json_string_d`, `log_error`, `log_pedantic`, `meta_data_fetch_all_tags()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_MESSAGES_TAGS`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_char_get()`.

5.452.1.39 void portal_endpoint_response (connection_t * con, chr_t * format, ...)

Generate a json-rpc 2.0 response to a portal request.

See also:

`json_vpack_ex()`

Note:

This function indents the json response if specified in the configuration, and also automatically decreases the reference count of any json object that was packed for the reply.

Parameters:

con a pointer to the connection object across which the portal response will be sent.

format a pointer to a format string specifying the construction of the json-rpc response.

... a variable arguments style list of parameters to be passed to the json packing function.

Returns:

This function returns no value.

Definition at line 162 of file endpoint.c.

References `con_write_st()`, `HTTP_ERROR_500`, `http_response_header()`, `json_decref_d`, `json_dumps_d`, `json_vpack_ex_d`, `log_pedantic`, `magma`, `ns_append()`, `ns_free()`, `ns_length_get()`, `NULLER`, `PLACER`, `magma_t::portal`, and `magma_t::web`.

Referenced by `portal_endpoint_ad()`, `portal_endpoint_alert_acknowledge()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_auth()`, `portal_endpoint_config_edit()`, `portal_endpoint_config_load()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_cookies()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_logout()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_folder_contacts_add()`, `portal_folder_contacts_remove()`, `portal_folder_mail_add()`, `portal_folder_mail_remove()`, `portal_meta()`, `portal_settings_changepass()`, and `portal_settings_identity()`.

5.452.1.40 void portal_endpoint_scrape (connection_t * con)

Note:

This function is not implemented and may be removed entirely in the future.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5767 of file endpoint.c.

References `JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_AD`, and `portal_validate_request()`.

5.452.1.41 void portal_endpoint_scrape_add (connection_t * con)

Note:

This function is not implemented and may be removed entirely in the future.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5751 of file endpoint.c.

References `JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_AD`, and `portal_validate_request()`.

5.452.1.42 void portal_endpoint_search (connection_t * con)

Definition at line 5729 of file endpoint.c.

References `con_write_st()`, `HTTP_ERROR_500`, `http_response_header()`, `PLACER`, `PORTAL_ENDPOINT_ERROR_SEARCH`, `st_aprint()`, `st_free()`, and `st_length_get()`.

5.452.1.43 void portal_endpoint_sort (void)

Sort the web portal JSON command table to be ready for binary searches. TODO: We use session locks to synchronize access to the user context. Should we also be using mailbox cluster locks? Cluster level locking might be appropriate for operations that delete data likes folders.remove and messages.remove.

Returns:

This function returns no value.

Definition at line 21 of file endpoint.c.

References command_t, portal_endpoint_compare(), and portal_methods.

Referenced by protocol_init().

5.452.1.44 attachment_t* portal_get_upload_attachment (connection_t * con)

Get a user's attachment to a message composition uploaded via multipart form data.

Note:**Parameters:**

path a pointer to a managed string containing the full /attach path specified in the client's http POST request.

Definition at line 6003 of file endpoint.c.

References attachment_t, composition_t, inx_find(), log_pedantic, M_TYPE_UINT64, mutex_lock(), mutex_unlock(), placer_t, tok_get_count_st(), tok_get_st(), multi_t::u64, uint64_conv_st(), and multi_t::val.

Referenced by portal_upload().

5.452.1.45 void portal_meta (connection_t * con)

Return information for a portal "meta" json-rpc request.

Note:

This function returns various pieces of information about the user such as their plan type, quota, etc.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5834 of file endpoint.c.

References con_addr_presentation(), json_object_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, log_pedantic, MAGMA_PORTAL_VERSION, MANAGEDBUF, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_META, portal_endpoint_response(), portal_validate_request(), and st_char_get().

5.452.1.46 void portal_settings_changepass (connection_t * con)

Change a user's password in response to a portal "settings.changepass" json-rpc request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5871 of file endpoint.c.

References `json_object_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `MAGMA_PORTAL_VERSION`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_SETTINGS_CHANGEPASS`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, and `st_length_get()`.

5.452.1.47 void portal_settings_identity (connection_t * con)

Return information for a portal "settings.identity" json-rpc request.

Note:

This function returns the full name, first name, last name, and website of the requested user.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5801 of file endpoint.c.

References `json_object_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `log_pedantic`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_SETTINGS_IDENTITY`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_char_get()`.

5.452.1.48 void portal_upload (connection_t * con)

Process uploaded attachments for messages composed in conjunction with the portal interface.

Definition at line 6074 of file endpoint.c.

References `attachment_t`, `con_localhost()`, `con_secure()`, `data`, `get_header_opt()`, `http_data_free()`, `http_data_header_parse_line()`, `HTTP_ERROR_401`, `HTTP_ERROR_403`, `HTTP_ERROR_404`, `HTTP_ERROR_405`, `HTTP_ERROR_500`, `HTTP_MERGED`, `HTTP_METHOD_POST`, `http_parse_context()`, `HTTP_PORTAL`, `http_print_301()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_error`, `log_pedantic`, `magma`, `multipart_get_boundary()`, `http_data_t::name`, `NULLER`, `pl_char_get()`, `pl_empty()`, `pl_length_get()`, `pl_shrink_before_characters()`, `pl_skip_characters()`, `pl_skip_to_characters()`, `PLACER`, `placer_t`, `magma_t::portal`, `portal_get_upload_attachment()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_import()`, `st_length_get()`, `str_tok_get_bl()`, `str_tok_get_count_bl()`, `http_data_t::value`, and `magma_t::web`.

Referenced by `http_response()`.

5.452.1.49 bool_t portal_validate_request (connection_t * con, int err_mask, chr_t * method, bool_t has_params, size_t nparams)

Verify that a session underlying a portal request has been authenticated.

Note:

If the session is not authenticated, an error message will be sent to the client, so the caller only needs to return from its function if this function returns false.

Parameters:

con a pointer to the connection object across which the portal request was made.

err_mask an optional error bitmask to be combined with any portal authentication error code.

method a pointer to a null-terminated string that will be used in an optional error logging message if it is not NULL.

has_params if true, return an error if the authenticated portal session supplied no method parameters; if false, return an error if the authenticated portal supplies params and they're not empty.

nparams if non-zero and *has_params* is also set, return an error if the portal session's parameters count does not match this value.

Returns:

Definition at line 64 of file endpoint.c.

References `json_object_size_d`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `log_pedantic`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_MODE`, `SESSION_STATE_AUTHENTICATED`, `st_char_get()`, and `st_length_get()`.

Referenced by `portal_endpoint_ad()`, `portal_endpoint_alert_acknowledge()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_config_edit()`, `portal_endpoint_config_load()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_cookies()`, `portal_endpoint_folders_add()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_remove()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_logout()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_endpoint_scrape()`, `portal_endpoint_scrape_add()`, `portal_meta()`, `portal_settings_changepass()`, and `portal_settings_identity()`.

5.453 src/web/portal/mail.c File Reference

```
#include "magma.h"
```

Functions

- [bool_t portal_outbound_checks](#) (uint64_t usernum, [stringer_t](#) *username, [stringer_t](#) *verification, [stringer_t](#) *from, size_t num_recipients, [stringer_t](#) *body_plain, [stringer_t](#) *body_html, [chr_t](#) **errmsg)
Perform a series of security-related checks on an outbound email message in a transport-independent manner.
- [bool_t portal_smtp_relay_message](#) ([stringer_t](#) *from, [inx_t](#) *to, [stringer_t](#) *data, size_t send_size, [chr_t](#) **errmsg)
Send (relay) a message composed by a user via a portal session.
- [stringer_t](#) * [portal_smtp_create_data](#) ([inx_t](#) *attachments, [stringer_t](#) *from, [inx_t](#) *to, [inx_t](#) *cc, [inx_t](#) *bcc, [stringer_t](#) *subject, [stringer_t](#) *body_plain, [stringer_t](#) *body_html)
Create the data of an outbound smtp message that will be specified with the smtp DATA command.
- [stringer_t](#) * [portal_smtp_merge_headers](#) ([inx_t](#) *headers, [stringer_t](#) *leading, [stringer_t](#) *trailing)
Merge a list of smtp message headers into a single string, preceded by the leading text and followed by the trailing text.

5.453.1 Function Documentation

5.453.1.1 [bool_t portal_outbound_checks](#) (uint64_t usernum, [stringer_t](#) * username, [stringer_t](#) * verification, [stringer_t](#) * from, size_t num_recipients, [stringer_t](#) * body_plain, [stringer_t](#) * body_html, [chr_t](#) ** errmsg)

Perform a series of security-related checks on an outbound email message in a transport-independent manner. [mail.c](#)

See also:

[smtp_data_outbound\(\)](#)

Note:

The checks include: Pattern matching for junk mail, authorization for the user to use the email address or domain specified in the From address, and finally, a virus check.

Parameters:

cred a credential structure obtained for the authenticated user's session.
usenum the numerical id of the user attempting to send the message.
from a managed string containing the email address specified as the From address.
nrecipients the number of recipients that will receive the sent message.
body_plain a managed string containing the plain text body of the message.
body_html a managed string containing the html body of the message.
errmsg the address of a pointer to a null-terminated string that will be set to a descriptive error message on failure.

Returns:

true if all security checks were passed or false otherwise.

Definition at line 24 of file mail.c.

References [pattern_check\(\)](#), [smtp_check_authorized_from\(\)](#), [smtp_check_transmit_quota\(\)](#), [smtp_fetch_authorization\(\)](#), and [virus_check\(\)](#).

Referenced by [portal_endpoint_messages_send\(\)](#).

5.453.1.2 **stringer_t* portal_smtp_create_data** (inx_t * *attachments*, stringer_t * *from*, inx_t * *to*, inx_t * *cc*, inx_t * *bcc*, stringer_t * *subject*, stringer_t * *body_plain*, stringer_t * *body_html*)

Create the data of an outbound smtp message that will be specified with the smtp DATA command.

Parameters:

attachments an optional inx holder containing a list of attachments to be included in the message.

from a managed string containing the email address sending the message.

to an inx holder containing a list of To: email recipients as managed strings.

cc an inx holder containing a list of 0 or more CC: recipients as managed strings.

bcc an inx holder containing a list of 0 or more BCC: recipients as managed strings.

subject a managed string containing the subject of the message.

body_plain an optional managed string containing the plain text body of the message.

body_html an optional managed string containing the html-formatted body of the message.

Returns:

NULL on failure or a managed string containing the packaged outbound smtp message data on success.

Definition at line 177 of file mail.c.

References `ar_append()`, `ar_free()`, `ar_length_get()`, `ARRAY_TYPE_POINTER`, `attachment_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_reset()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `mail_mime_encode_part()`, `mail_mime_generate_boundary()`, `mail_mime_get_smtp_envelope()`, `st_free()`, and `st_merge`.

Referenced by `portal_endpoint_messages_send()`.

5.453.1.3 **stringer_t* portal_smtp_merge_headers** (inx_t * *headers*, stringer_t * *leading*, stringer_t * *trailing*)

Merge a list of smtp message headers into a single string, preceded by the leading text and followed by the trailing text.

Parameters:

headers an inx holder containing a collection of header string data to be merged together.

leading a managed string containing text that will lead each header line.

trailing a managed string containing text that will trail each header line.

Returns:

NULL on failure or a managed string containing the merged headers on success.

Definition at line 301 of file mail.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `st_alloc()`, `st_cleanup`, and `st_merge`.

Referenced by `mail_mime_get_smtp_envelope()`.

5.453.1.4 **bool_t portal_smtp_relay_message** (stringer_t * *from*, inx_t * *to*, stringer_t * *data*, size_t *send_size*, chr_t ** *errmsg*)

Send (relay) a message composed by a user via a portal session.

See also:

[smtp_relay_message\(\)](#) - a lot of logic borrowed from here.

Parameters:

- from* a pointer to a managed string containing the email address specified as the From address.
- to* a pointer to a managed string containing the destination email address of the message.
- data* a pointer to a managed string containing the raw data of the mail message.
- send_size* if greater than 0, specify the optional SIZE parameter to the MAIL FROM command.
- errmsg* the address of a pointer to a null-terminated string that will be set to a descriptive error message on failure.

Returns:

true if the mail message was sent successfully, or false otherwise.

Definition at line 86 of file mail.c.

References `client_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `smtp_client_close()`, `smtp_client_connect()`, `smtp_client_send_data()`, `smtp_client_send_helo()`, `smtp_client_send_mailfrom()`, and `smtp_client_send_rcptto()`.

Referenced by `portal_endpoint_messages_send()`.

5.454 src/web/portal/methods.h File Reference

Variables

- [command_t portal_methods](#) []

5.454.1 Variable Documentation

5.454.1.1 [command_t portal_methods](#) []

Definition at line 22 of file methods.h.

Referenced by [portal_endpoint\(\)](#), and [portal_endpoint_sort\(\)](#).

5.455 src/web/portal/portal.c File Reference

```
#include "magma.h"
```

Functions

- void [portal_print_login](#) ([connection_t](#) *con, [chr_t](#) *message)
Display the http portal login page.
- void [portal_process](#) ([connection_t](#) *con)
Process a connection to the web portal.

5.455.1 Function Documentation

5.455.1.1 void [portal_print_login](#) ([connection_t](#) * con, [chr_t](#) * message)

Display the http portal login page. [portal.c](#)

Note:

The portal login page can be found in 'portal/login.template'

Parameters:

con a pointer to the connection object of the client requesting the portal login page.

message a pointer to a null-terminated string that will be displayed to the user in the login page as the contents of the node with the id of "message" in the portal login template.

Returns:

This function returns no value.

Definition at line 18 of file portal.c.

References [con_write_st\(\)](#), [http_page_t::content](#), [http_page_t::doc_obj](#), [http_page_free\(\)](#), [http_page_get\(\)](#), [http_print_500\(\)](#), [http_response_header\(\)](#), [st_free\(\)](#), [st_length_get\(\)](#), [http_content_t::type](#), [xml_dump_doc\(\)](#), [xml_node_set_content\(\)](#), [xml_xpath_eval\(\)](#), and [http_page_t::xpath_ctx](#).

Referenced by [portal_process\(\)](#).

5.455.1.2 void [portal_process](#) ([connection_t](#) * con)

Process a connection to the web portal.

Note:

If magma.web.portal.safeguard is set, the user will be redirected to a secure login.

Parameters:

con a pointer to the connection of the http client requesting the portal.

Returns:

This function returns no value.

Definition at line 57 of file portal.c.

References `con_localhost()`, `con_secure()`, `http_parse_context()`, `http_print_301()`, `magma`, `PLACER`, `magma_t::portal`, `portal_print_login()`, and `magma_t::web`.

Referenced by `http_response()`.

5.456 src/web/portal/portal.h File Reference

Defines

- #define [MAGMA_PORTAL_VERSION](#) "1.02"

Enumerations

- enum {
[JSON_RPC_2_ERROR_PARSE_MALFORMED](#) = -32700, [JSON_RPC_2_ERROR_PARSE_ENCODING](#) = -32701, [JSON_RPC_2_ERROR_PARSE_CHAR](#) = -32702, [JSON_RPC_2_ERROR_SERVER_REQUEST](#) = -32600,
[JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL](#) = -32601, [JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS](#) = -32602, [JSON_RPC_2_ERROR_SERVER_INTERNAL](#) = -32603, [JSON_RPC_2_ERROR_APPLICATION](#) = -32500,
[JSON_RPC_2_ERROR_SYSTEM](#) = -32400, [JSON_RPC_2_ERROR_TRANSPORT](#) = -32300 }
- enum {
[PORTAL_ENDPOINT_ACTION_INVALID](#) = -1, [PORTAL_ENDPOINT_ACTION_ADD](#) = 1, [PORTAL_ENDPOINT_ACTION_REMOVE](#) = 2, [PORTAL_ENDPOINT_ACTION_REPLACE](#) = 3,
[PORTAL_ENDPOINT_ACTION_LIST](#) = 4 }
- enum {
[PORTAL_ENDPOINT_CONTEXT_INVALID](#) = -1, [PORTAL_ENDPOINT_CONTEXT_MAIL](#) = 1, [PORTAL_ENDPOINT_CONTEXT_CONTACTS](#) = 2, [PORTAL_ENDPOINT_CONTEXT_SETTINGS](#) = 3,
[PORTAL_ENDPOINT_CONTEXT_HELP](#) = 4 }
- enum {
[PORTAL_ENDPOINT_MESSAGE_META](#) = 1, [PORTAL_ENDPOINT_MESSAGE_SOURCE](#) = 2, [PORTAL_ENDPOINT_MESSAGE_SECURITY](#) = 4, [PORTAL_ENDPOINT_MESSAGE_SERVER](#) = 8,
[PORTAL_ENDPOINT_MESSAGE_HEADER](#) = 16, [PORTAL_ENDPOINT_MESSAGE_BODY](#) = 32, [PORTAL_ENDPOINT_MESSAGE_ATTACHMENTS](#) = 64, [PORTAL_ENDPOINT_MESSAGE_INFO](#) = 128 }
- enum {
[PORTAL_ENDPOINT_ERROR_MODE](#) = 10000, [PORTAL_ENDPOINT_ERROR_REFERENCE](#) = 10100, [PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION](#) = 10200, [PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD](#) = 10300,
[PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION](#) = 10300, [PORTAL_ENDPOINT_ERROR_SYSTEM_FLAG](#) = 10400, [PORTAL_ENDPOINT_ERROR_READ](#) = 10500 }
- enum {
[PORTAL_ENDPOINT_ERROR_NONE](#) = 0, [PORTAL_ENDPOINT_ERROR_AUTH](#), [PORTAL_ENDPOINT_ERROR_COOKIES](#), [PORTAL_ENDPOINT_ERROR_AD](#),
[PORTAL_ENDPOINT_ERROR_ALERT_LIST](#), [PORTAL_ENDPOINT_ERROR_ALERT_ACKNOWLEDGE](#), [PORTAL_ENDPOINT_ERROR_ALIASES](#), [PORTAL_ENDPOINT_ERROR_SCRAPE_ADD](#),
[PORTAL_ENDPOINT_ERROR_SCRAPE](#), [PORTAL_ENDPOINT_ERROR_ATTACHMENTS_ADD](#), [PORTAL_ENDPOINT_ERROR_ATTACHMENTS_PROGRESS](#), [PORTAL_ENDPOINT_ERROR_ATTACHMENTS_REMOVE](#),
[PORTAL_ENDPOINT_ERROR_CONFIG_LOAD](#), [PORTAL_ENDPOINT_ERROR_CONFIG_EDIT](#), [PORTAL_ENDPOINT_ERROR_CONTACTS_ADD](#), [PORTAL_ENDPOINT_ERROR_CONTACTS_EDIT](#),
[PORTAL_ENDPOINT_ERROR_CONTACTS_LIST](#), [PORTAL_ENDPOINT_ERROR_CONTACTS_LOAD](#), [PORTAL_ENDPOINT_ERROR_CONTACTS_MOVE](#), [PORTAL_ENDPOINT_ERROR_CONTACTS_COPY](#),
[PORTAL_ENDPOINT_ERROR_CONTACTS_REMOVE](#), [PORTAL_ENDPOINT_ERROR_FOLDERS_ADD](#), [PORTAL_ENDPOINT_ERROR_FOLDERS_LIST](#), [PORTAL_ENDPOINT_ERROR_FOLDERS_TAGS](#),
[PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE](#), [PORTAL_ENDPOINT_ERROR_FOLDERS_RENAME](#), [PORTAL_ENDPOINT_ERROR_MESSAGES_COMPOSE](#), [PORTAL_ENDPOINT_ERROR_MESSAGES_COPY](#),
[PORTAL_ENDPOINT_ERROR_MESSAGES_FLAG](#), [PORTAL_ENDPOINT_ERROR_MESSAGES_LIST](#), [PORTAL_ENDPOINT_ERROR_MESSAGES_LOAD](#), [PORTAL_ENDPOINT_ERROR_MESSAGES_MOVE](#),

```

PORTAL_ENDPOINT_ERROR_MESSAGES_REMOVE,    PORTAL_ENDPOINT_ERROR_MESSAGES_SEND,    PORTAL_
ENDPOINT_ERROR_MESSAGES_TAG, PORTAL_ENDPOINT_ERROR_MESSAGES_TAGS,
PORTAL_ENDPOINT_ERROR_META, PORTAL_ENDPOINT_ERROR_SEARCH, PORTAL_ENDPOINT_ERROR_SETTINGS_
IDENTITY, PORTAL_ENDPOINT_ERROR_SETTINGS_CHANGEPASS,
PORTAL_ENDPOINT_ERROR_LOGOUT }

```

Functions

- json_t * portal_config_collection (user_config_t *collection)
config.c
- json_t * portal_config_entry (user_config_entry_t *entry)
- json_t * portal_config_entry_flags (user_config_entry_t *entry)
Get a user config entry's flags as a json object.
- json_t * portal_contact_detail_flags (contact_detail_t *detail)
contacts.c
- json_t * portal_contact_details (contact_t *contact)
Retrieve all the details about a contact entry as a json object.
- void portal_endpoint (connection_t *con)
endpoint.c
- void portal_endpoint_ad (connection_t *con)
Return advertising information for a portal "ad" json-rpc request.
- void portal_endpoint_alert_acknowledge (connection_t *con)
Process user alert message acknowledgements in response to an "alert.acknowledge" json-rpc portal request.
- void portal_endpoint_alert_list (connection_t *con)
Get the list of a user's alert messages in response to an "alert.list" json-rpc portal request.
- void portal_endpoint_aliases (connection_t *con)
Return the list of a user account's aliases in response to an "aliases" json-rpc portal request.
- void portal_endpoint_attachments_add (connection_t *con)
Add an attachment to a message being composed in response to an "attachments.add" json-rpc portal request.
- void portal_endpoint_attachments_progress (connection_t *con)
- void portal_endpoint_attachments_remove (connection_t *con)
Remove an attachment from a message being composed in response to an "attachments.remove" json-rpc portal request.
- void portal_endpoint_auth (connection_t *con)
Obtain credentials and log a user into portal session in response to an "auth" json-rpc portal request.
- int_t portal_endpoint_compare (const void *compare, const void *command)
Internal bsearch() comparison function for dispatching the proper portal handler.
- void portal_endpoint_config_edit (connection_t *con)
Retrieve all of a user's config options and return them to the remote client in response to a portal "contacts.load" json-rpc request.

- void [portal_endpoint_config_load](#) ([connection_t](#) *con)
Retrieve all of a user's config options and return them to the remote client in response to a portal "contacts.load" json-rpc request.
- void [portal_endpoint_contacts_add](#) ([connection_t](#) *con)
Add a contact in response to a "contacts.add" json-rpc portal request.
- void [portal_endpoint_contacts_copy](#) ([connection_t](#) *con)
- void [portal_endpoint_contacts_edit](#) ([connection_t](#) *con)
- void [portal_endpoint_contacts_list](#) ([connection_t](#) *con)
List a user's contacts in response to a "contacts.list" json-rpc portal request.
- void [portal_endpoint_contacts_load](#) ([connection_t](#) *con)
Return information for a portal "contacts.load" json-rpc request.
- void [portal_endpoint_contacts_move](#) ([connection_t](#) *con)
Move a user's contact entry to another contacts folder in response to a "contacts.move" json-rpc portal request.
- void [portal_endpoint_contacts_remove](#) ([connection_t](#) *con)
Remove a user's contact entry in response to a "contacts.remove" json-rpc portal request.
- void [portal_endpoint_cookies](#) ([connection_t](#) *con)
Return whether or not a user is using cookies, in response to a "cookies" json-rpc portal request.
- void [portal_endpoint_error](#) ([connection_t](#) *con, [int_t](#) http_code, [int_t](#) error_code, [chr_t](#) *message)
Return a json-rpc error response to the remote client.
- void [portal_endpoint_folders_add](#) ([connection_t](#) *con)
Create a new folder in response to a "folders.add" json-rpc portal request.
- void [portal_endpoint_folders_list](#) ([connection_t](#) *con)
Return a list of a user's folders in response to a "folders.list" json-rpc portal request.
- void [portal_endpoint_folders_remove](#) ([connection_t](#) *con)
Remove a user's folder in response to a json-rpc "folders.remove" portal request.
- void [portal_endpoint_folders_rename](#) ([connection_t](#) *con)
Rename a user's folder in response to a json-rpc "folders.rename" portal request.
- void [portal_endpoint_folders_tags](#) ([connection_t](#) *con)
- void [portal_endpoint_logout](#) ([connection_t](#) *con)
Log a user out of a portal session in response to a "logout" json-rpc portal request.
- void [portal_endpoint_messages_compose](#) ([connection_t](#) *con)
Compose a new message in response to a "messages.compose" json-rpc portal request.
- void [portal_endpoint_messages_copy](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_flag](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_list](#) ([connection_t](#) *con)
Retrieve a list of the user's messages in response to a json-rpc "messages.list" portal request.
- void [portal_endpoint_messages_load](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_move](#) ([connection_t](#) *con)
- void [portal_endpoint_messages_remove](#) ([connection_t](#) *con)

Remove a user's mail message in response to a "messages.remove" json-rpc portal request.

- void `portal_endpoint_messages_tags` (`connection_t` *con)

Get all of the message tags used by a specified user in response to a "messages.tags" json-rpc portal request.

- void `portal_endpoint_messages_tag` (`connection_t` *con)

Perform a tag operation on a user's message, in response to a "messages.tag" json-rpc portal request.

- void `portal_endpoint_response` (`connection_t` *con, `chr_t` *format,...)

Generate a json-rpc 2.0 response to a portal request.

- void `portal_endpoint_scrape` (`connection_t` *con)
- void `portal_endpoint_scrape_add` (`connection_t` *con)
- void `portal_endpoint_search` (`connection_t` *con)
- void `portal_settings_identity` (`connection_t` *con)

Return information for a portal "settings.identity" json-rpc request.

- void `portal_meta` (`connection_t` *con)

Return information for a portal "meta" json-rpc request.

- void `portal_endpoint_sort` (void)

Sort the web portal JSON command table to be ready for binary searches.

- void `portal_upload` (`connection_t` *con)

Process uploaded attachments for messages composed in conjunction with the portal interface.

- void `portal_endpoint_messages_send` (`connection_t` *con)

Send a composed message in response to a "messages.send" json-rpc portal request.

- void `portal_debug` (`connection_t` *con)

A portal debug function that will be disabled and/or deleted completely in production.

- `bool_t` `portal_outbound_checks` (`uint64_t` usernum, `stringer_t` *username, `stringer_t` *verification, `stringer_t` *from, `size_t` num_recipients, `stringer_t` *body_plain, `stringer_t` *body_html, `chr_t` **errmsg)

mail.c

- `bool_t` `portal_smtp_relay_message` (`stringer_t` *from, `inx_t` *to, `stringer_t` *data, `size_t` send_size, `chr_t` **errmsg)

Send (relay) a message composed by a user via a portal session.

- `stringer_t` * `portal_smtp_create_data` (`inx_t` *attachments, `stringer_t` *from, `inx_t` *to, `inx_t` *cc, `inx_t` *bcc, `stringer_t` *subject, `stringer_t` *body_plain, `stringer_t` *body_html)

Create the data of an outbound smtp message that will be specified with the smtp DATA command.

- `stringer_t` * `portal_smtp_merge_headers` (`inx_t` *headers, `stringer_t` *leading, `stringer_t` *trailing)

Merge a list of smtp message headers into a single string, preceded by the leading text and followed by the trailing text.

- `json_t` * `portal_message_flags_array` (`meta_message_t` *meta)

flags.c

- `json_t` * `portal_message_tags_array` (`meta_message_t` *meta)

- `int_t` `portal_parse_flags` (`json_t` *array, `uint32_t` *flags)

- void `portal_folder_contacts_add` (`connection_t` *con, `stringer_t` *name, `uint64_t` parent)

folders.c

- void [portal_folder_contacts_remove](#) ([connection_t](#) *con, [uint64_t](#) foldernum)
Remove a user's contacts folder.
- void [portal_folder_mail_add](#) ([connection_t](#) *con, [stringer_t](#) *name, [uint64_t](#) parent)
Add a new mail folder for a user.
- void [portal_folder_mail_remove](#) ([connection_t](#) *con, [uint64_t](#) foldernum)
Remove a user's mail folder.
- [json_t](#) * [portal_message_attachments](#) ([meta_message_t](#) *meta, [mail_message_t](#) *data)
messages.c
- [json_t](#) * [portal_message_body](#) ([meta_message_t](#) *meta, [mail_message_t](#) *data)
- [json_t](#) * [portal_message_header](#) ([meta_message_t](#) *meta, [mail_message_t](#) *data)
Build the "header" section response to a rpc-json "messages.load" request.
- [json_t](#) * [portal_message_meta](#) ([meta_message_t](#) *meta)
- [json_t](#) * [portal_message_security](#) ([meta_message_t](#) *meta)
- [json_t](#) * [portal_message_server](#) ([meta_message_t](#) *meta)
- [json_t](#) * [portal_message_source](#) ([meta_message_t](#) *meta)
- [json_t](#) * [portal_message_info](#) ([meta_message_t](#) *meta)
Build the "info" section response to a rpc-json "messages.load" request.
- [inx_t](#) * [portal_parse_json_str_array](#) ([json_t](#) *json, [size_t](#) *nout)
parse.c
- [int_t](#) [portal_parse_action](#) ([stringer_t](#) *action)
- [int_t](#) [portal_parse_context](#) ([stringer_t](#) *context)
Parse the context of a requested folder.
- [int_t](#) [portal_parse_sections](#) ([json_t](#) *array, [uint32_t](#) *sections)
Parse the json-rpc messages.load parameter "section" from an array of strings into a bitmask of section flags.
- void [portal_print_login](#) ([connection_t](#) *con, [chr_t](#) *message)
portal.c
- void [portal_process](#) ([connection_t](#) *con)
Process a connection to the web portal.

5.456.1 Define Documentation

5.456.1.1 #define MAGMA_PORTAL_VERSION "1.02"

Definition at line 11 of file portal.h.

Referenced by [portal_meta\(\)](#), and [portal_settings_changepass\(\)](#).

5.456.2 Enumeration Type Documentation

5.456.2.1 anonymous enum

Enumerator:

JSON_RPC_2_ERROR_PARSE_MALFORMED
JSON_RPC_2_ERROR_PARSE_ENCODING
JSON_RPC_2_ERROR_PARSE_CHAR
JSON_RPC_2_ERROR_SERVER_REQUEST
JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL
JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS
JSON_RPC_2_ERROR_SERVER_INTERNAL
JSON_RPC_2_ERROR_APPLICATION
JSON_RPC_2_ERROR_SYSTEM
JSON_RPC_2_ERROR_TRANSPORT

Definition at line 14 of file portal.h.

5.456.2.2 anonymous enum

Enumerator:

PORTAL_ENDPOINT_ACTION_INVALID
PORTAL_ENDPOINT_ACTION_ADD
PORTAL_ENDPOINT_ACTION_REMOVE
PORTAL_ENDPOINT_ACTION_REPLACE
PORTAL_ENDPOINT_ACTION_LIST

Definition at line 27 of file portal.h.

5.456.2.3 anonymous enum

Enumerator:

PORTAL_ENDPOINT_CONTEXT_INVALID
PORTAL_ENDPOINT_CONTEXT_MAIL
PORTAL_ENDPOINT_CONTEXT_CONTACTS
PORTAL_ENDPOINT_CONTEXT_SETTINGS
PORTAL_ENDPOINT_CONTEXT_HELP

Definition at line 35 of file portal.h.

5.456.2.4 anonymous enum

Enumerator:

PORTAL_ENDPOINT_MESSAGE_META
PORTAL_ENDPOINT_MESSAGE_SOURCE
PORTAL_ENDPOINT_MESSAGE_SECURITY
PORTAL_ENDPOINT_MESSAGE_SERVER

PORTAL_ENDPOINT_MESSAGE_HEADER
PORTAL_ENDPOINT_MESSAGE_BODY
PORTAL_ENDPOINT_MESSAGE_ATTACHMENTS
PORTAL_ENDPOINT_MESSAGE_INFO

Definition at line 43 of file portal.h.

5.456.2.5 anonymous enum

Enumerator:

PORTAL_ENDPOINT_ERROR_MODE
PORTAL_ENDPOINT_ERROR_REFERENCE
PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION
PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD
PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION
PORTAL_ENDPOINT_ERROR_SYSTEM_FLAG
PORTAL_ENDPOINT_ERROR_READ

Definition at line 54 of file portal.h.

5.456.2.6 anonymous enum

Enumerator:

PORTAL_ENDPOINT_ERROR_NONE
PORTAL_ENDPOINT_ERROR_AUTH
PORTAL_ENDPOINT_ERROR_COOKIES
PORTAL_ENDPOINT_ERROR_AD
PORTAL_ENDPOINT_ERROR_ALERT_LIST
PORTAL_ENDPOINT_ERROR_ALERT_ACKNOWLEDGE
PORTAL_ENDPOINT_ERROR_ALIASES
PORTAL_ENDPOINT_ERROR_SCRAPE_ADD
PORTAL_ENDPOINT_ERROR_SCRAPE
PORTAL_ENDPOINT_ERROR_ATTACHMENTS_ADD
PORTAL_ENDPOINT_ERROR_ATTACHMENTS_PROGRESS
PORTAL_ENDPOINT_ERROR_ATTACHMENTS_REMOVE
PORTAL_ENDPOINT_ERROR_CONFIG_LOAD
PORTAL_ENDPOINT_ERROR_CONFIG_EDIT
PORTAL_ENDPOINT_ERROR_CONTACTS_ADD
PORTAL_ENDPOINT_ERROR_CONTACTS_EDIT
PORTAL_ENDPOINT_ERROR_CONTACTS_LIST
PORTAL_ENDPOINT_ERROR_CONTACTS_LOAD
PORTAL_ENDPOINT_ERROR_CONTACTS_MOVE
PORTAL_ENDPOINT_ERROR_CONTACTS_COPY
PORTAL_ENDPOINT_ERROR_CONTACTS_REMOVE
PORTAL_ENDPOINT_ERROR_FOLDERS_ADD

PORTAL_ENDPOINT_ERROR_FOLDERS_LIST
PORTAL_ENDPOINT_ERROR_FOLDERS_TAGS
PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE
PORTAL_ENDPOINT_ERROR_FOLDERS_RENAME
PORTAL_ENDPOINT_ERROR_MESSAGES_COMPOSE
PORTAL_ENDPOINT_ERROR_MESSAGES_COPY
PORTAL_ENDPOINT_ERROR_MESSAGES_FLAG
PORTAL_ENDPOINT_ERROR_MESSAGES_LIST
PORTAL_ENDPOINT_ERROR_MESSAGES_LOAD
PORTAL_ENDPOINT_ERROR_MESSAGES_MOVE
PORTAL_ENDPOINT_ERROR_MESSAGES_REMOVE
PORTAL_ENDPOINT_ERROR_MESSAGES_SEND
PORTAL_ENDPOINT_ERROR_MESSAGES_TAG
PORTAL_ENDPOINT_ERROR_MESSAGES_TAGS
PORTAL_ENDPOINT_ERROR_META
PORTAL_ENDPOINT_ERROR_SEARCH
PORTAL_ENDPOINT_ERROR_SETTINGS_IDENTITY
PORTAL_ENDPOINT_ERROR_SETTINGS_CHANGEPASS
PORTAL_ENDPOINT_ERROR_LOGOUT

Definition at line 64 of file portal.h.

5.456.3 Function Documentation

5.456.3.1 json_t* portal_config_collection (user_config_t *collection)

config.c config.c

Parameters:

collection a pointer to a user config object that contains all the config options pairs to be serialized.

Returns:

NULL on failure, or a pointer to a json object containing the collection of user config options on success.

Definition at line 62 of file config.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `json_decref_d`, `json_object_d`, `json_object_set_new_d`, `log_error`, `log_options`, `M_LOG_CRITICAL`, `portal_config_entry()`, `st_char_get()`, `USER_CONF_STATUS_CRITICAL`, and `user_config_entry_t`.

Referenced by `portal_endpoint_config_load()`.

5.456.3.2 json_t* portal_config_entry (user_config_entry_t *entry)

Definition at line 35 of file config.c.

References `json_decref_d`, `json_pack_ex_d`, `log_pedantic`, `portal_config_entry_flags()`, `st_char_get()`, and `st_length_int()`.

Referenced by `portal_config_collection()`.

5.456.3.3 json_t* portal_config_entry_flags (user_config_entry_t * entry)

Get a user config entry's flags as a json object.

Parameters:

entry a pointer to the user config entry object to have its flags serialized.

Returns:

NULL on failure or a pointer to a json object storing the user's flags on success.

Definition at line 15 of file config.c.

References json_array_append_new_d, json_array_d, json_decref_d, json_string_d, and USER_CONF_STATUS_CRITICAL.

Referenced by portal_config_entry().

5.456.3.4 json_t* portal_contact_detail_flags (contact_detail_t * detail)

contacts.c contacts.c

Pack a json array representing the flags associated with a user contact entry detail.

Parameters:

detail a pointer to the user contact detail to have its flags queried.

Returns:

NULL on failure or a pointer to a json array containing strings describing the specified contact detail's flags.

Definition at line 16 of file contacts.c.

References CONTACT_DETAIL_FLAG_CRITICAL, json_array_append_new_d, json_array_d, and json_string_d.

Referenced by portal_contact_details().

5.456.3.5 json_t* portal_contact_details (contact_t * contact)

Retrieve all the details about a contact entry as a json object.

Parameters:

contact a pointer to the contact object whose details will be retrieved.

Returns:

a pointer to a json object containing the details of the specified contact entry.

Definition at line 35 of file contacts.c.

References contact_detail_t, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), json_array_d, json_decref_d, json_object_set_new_d, json_pack_ex_d, log_error, log_pedantic, portal_contact_detail_flags(), and st_char_get().

Referenced by portal_endpoint_contacts_load().

5.456.3.6 void portal_debug (connection_t * con)

A portal debug function that will be disabled and/or deleted completely in production.

Note:

The debug output is displayed LOCALLY in the Magma console.

Definition at line 6254 of file endpoint.c.

5.456.3.7 void portal_endpoint (connection_t * con)

[endpoint.c](#) [endpoint.c](#)

Parameters:

con the connection object corresponding to the web client making the request.

Returns:

This function returns no value.

Definition at line 5913 of file endpoint.c.

References `command`, `command_t`, `con_localhost()`, `con_secure()`, `connection_t`, `HTTP_ERROR_500`, `HTTP_MERGED`, `http_parse_context()`, `HTTP_PORTAL`, `http_print_301()`, `http_response_header()`, `json_integer_value_d`, `json_loads_d`, `json_object_get_d`, `JSON_RPC_2_ERROR_PARSE_MALFORMED`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL`, `JSON_RPC_2_ERROR_SERVER_REQUEST`, `json_string_value_d`, `json_type_string_d`, `log_pedantic`, `magma`, `ns_length_get()`, `PLACER`, `magma_t::portal`, `portal_endpoint_compare()`, `portal_endpoint_error()`, `portal_methods`, `sess_create()`, `st_char_get()`, `uint64_conv_ns()`, and `magma_t::web`.

Referenced by `http_response()`.

5.456.3.8 void portal_endpoint_ad (connection_t * con)

Return advertising information for a portal "ad" json-rpc request. ^ finished / work area v ^ work area / stubs v

Note:

This function is not implemented and may be removed entirely in the future.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2888 of file endpoint.c.

References `json_object_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `log_pedantic`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_AD`, `portal_endpoint_response()`, and `portal_validate_request()`.

5.456.3.9 void portal_endpoint_alert_acknowledge (connection_t * con)

Process user alert message acknowledgements in response to an "alert.acknowledge" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 444 of file endpoint.c.

References `count`, `json_array_get_d`, `json_array_size_d`, `json_integer_value_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `meta_data_acknowledge_alert()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_ALERT_ACKNOWLEDGE`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, `st_length_get()`, `tran_commit()`, `tran_rollback()`, and `tran_start()`.

5.456.3.10 void portal_endpoint_alert_list (connection_t * con)

Get the list of a user's alert messages in response to an "alert.list" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 396 of file endpoint.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `log_pedantic`, `meta_alert_t`, `meta_data_fetch_alerts()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_ALERT_LIST`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_char_get()`.

5.456.3.11 void portal_endpoint_aliases (connection_t * con)

Return the list of a user account's aliases in response to an "aliases" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 514 of file endpoint.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `json_true_d`, `log_pedantic`, `meta_alias_t`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_ALIASES`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_char_get()`.

5.456.3.12 void portal_endpoint_attachments_add (connection_t * con)

Add an attachment to a message being composed in response to an "attachments.add" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2002 of file endpoint.c.

References `attachment_t`, `composition_t`, `inx_find()`, `inx_insert()`, `json_object_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_error`, `log_pedantic`, `M_TYPE_UINT64`,

mm_alloc(), mutex_lock(), mutex_unlock(), ns_length_get(), portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ATTACHMENTS_ADD, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), sess_release_attachment(), st_char_get(), st_import(), st_length_get(), multi_t::u64, and multi_t::val.

5.456.3.13 void portal_endpoint_attachments_progress (connection_t * con)

Definition at line 5778 of file endpoint.c.

References con_write_st(), HTTP_ERROR_500, http_response_header(), PLACER, PORTAL_ENDPOINT_ERROR_ATTACHMENTS_PROGRESS, st_aprint(), st_free(), and st_length_get().

5.456.3.14 void portal_endpoint_attachments_remove (connection_t * con)

Remove an attachment from a message being composed in response to an "attachments.remove" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2080 of file endpoint.c.

References attachment_t, composition_t, inx_delete(), inx_find(), JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_error, log_pedantic, M_TYPE_UINT64, mutex_lock(), mutex_unlock(), portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ATTACHMENTS_REMOVE, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.456.3.15 void portal_endpoint_auth (connection_t * con)

Obtain credentials and log a user into portal session in response to an "auth" json-rpc portal request.

Parameters:

con a pointer to the connection object of the user attempting to log in.

Returns:

This function returns no value.

Definition at line 194 of file endpoint.c.

References auth_free(), AUTH_LOCK_ABUSE, AUTH_LOCK_ADMIN, AUTH_LOCK_EXPIRED, AUTH_LOCK_INACTIVITY, AUTH_LOCK_USER, auth_login(), cache_increment(), con_addr_presentation(), con_addr_subnet(), con_localhost(), con_secure(), HTTP_COOKIE_SET, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, auth_t::keys, lock_get(), lock_release(), auth_t::locked, log_info, log_pedantic, MANAGEDBUF, auth_t::master, meta_get(), META_GET_ALIASES, META_GET_CONTACTS, META_GET_FOLDERS, META_GET_KEYS, META_GET_MESSAGES, meta_inx_remove(), META_PROTOCOL_WEB, META_USER_ENCRYPT_DATA, meta_user_t, META_USER_TLS, NULLER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_AUTH, PORTAL_ENDPOINT_ERROR_MODE, portal_endpoint_response(), auth_t::salt, auth_t::seasoning, SESSION_STATE_AUTHENTICATED, SESSION_STATE_NEUTRAL, st_char_get(), st_length_int(), st_populated, st_quick(), auth_t::status, time_datestamp(), auth_t::tokens, auth_t::username, auth_t::usernum, and auth_t::verification.

5.456.3.16 int_t portal_endpoint_compare (const void * compare, const void * command)

Internal bsearch() comparison function for dispatching the proper portal handler.

Parameters:

compare a pointer to a `command_t` object with a portal method name for string comparison.

command a pointer to a `command_t` object with a portal method name for string comparison.

Returns:

Definition at line 36 of file `endpoint.c`.

References `command_t`, `PLACER`, and `st_cmp_ci_eq()`.

Referenced by `portal_endpoint()`, and `portal_endpoint_sort()`.

5.456.3.17 void portal_endpoint_config_edit (connection_t * con)

Retrieve all of a user's config options and return them to the remote client in response to a portal "contacts.load" json-rpc request.

Note:

Requires a user to be authenticated; failure will be communicated to the client via a json response.

Parameters:

con a pointer to a connection object over which the json response will be sent.

Returns:

This function returns no value.

HIGH: We aren't checking/validating the config entries! And we should probably do our own duplication checks to avoid placing unnecessary load on the database.

Definition at line 2739 of file `endpoint.c`.

References `content`, `json_object_iter_d`, `json_object_iter_key_d`, `json_object_iter_next_d`, `json_object_iter_value_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `json_string_value_d`, `NULLER`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONFIG_EDIT`, `portal_endpoint_response()`, `portal_validate_request()`, `status`, `user_config_create()`, `user_config_edit()`, `user_config_free()`, and `user_config_t`.

5.456.3.18 void portal_endpoint_config_load (connection_t * con)

Retrieve all of a user's config options and return them to the remote client in response to a portal "contacts.load" json-rpc request.

Note:

Requires a user to be authenticated; failure will be communicated to the client via a json response.

Parameters:

con a pointer to a connection object over which the json response will be sent.

Returns:

This function returns no value.

Definition at line 2798 of file `endpoint.c`.

References `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `portal_config_collection()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONFIG_LOAD`, `portal_endpoint_response()`, `portal_validate_request()`, `user_config_create()`, `user_config_free()`, and `user_config_t`.

5.456.3.19 void portal_endpoint_contacts_add (connection_t * con)

Add a contact in response to a "contacts.add" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

HIGH: We aren't checking/validating the contact details! And we should probably do our own duplication checks to avoid placing unnecessary load on the database.

Definition at line 2546 of file endpoint.c.

References contact_create(), contact_edit(), contact_free(), contact_t, contact_validate_detail(), contact_validate_name(), content, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_insert(), json_object_get_d, json_object_iter_d, json_object_iter_key_d, json_object_iter_next_d, json_object_iter_value_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_string_value_d, json_type_string_d, json_unpack_ex_d, log_pedantic, M_TYPE_UINT64, magma_folder_find_number(), meta_user_rlock(), meta_user_unlock(), meta_user_wlock(), NULLER, OBJECT_CONTACTS, PLACER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION, PORTAL_ENDPOINT_ERROR_CONTACTS_ADD, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), sess_serial_check(), st_char_get(), st_cmp_ci_eq(), st_cmp_cs_eq(), st_length_get(), status, multi_t::u64, and multi_t::val.

5.456.3.20 void portal_endpoint_contacts_copy (connection_t * con)**Note:**

If the source and target folder are equal, a copy will be made with a different name.

Definition at line 2252 of file endpoint.c.

References contact_create(), contact_detail_t, contact_edit(), contact_find_number(), contact_free(), contact_t, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_insert(), JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_error, log_pedantic, M_TYPE_UINT64, magma_folder_find_number(), meta_user_rlock(), meta_user_unlock(), meta_user_wlock(), OBJECT_CONTACTS, PLACER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION, PORTAL_ENDPOINT_ERROR_CONTACTS_COPY, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), sess_serial_check(), st_char_get(), st_cleanup, st_length_get(), st_merge, multi_t::u64, and multi_t::val.

5.456.3.21 void portal_endpoint_contacts_edit (connection_t * con)

HIGH: We aren't checking/validating the contact details! And we should probably do our own duplication checks to avoid placing unnecessary load on the database.

Definition at line 2662 of file endpoint.c.

References contact_edit(), contact_find_number(), contact_t, content, json_object_iter_d, json_object_iter_key_d, json_object_iter_next_d, json_object_iter_value_d, json_object_size_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_string_value_d, json_unpack_ex_d, log_pedantic, magma_folder_find_number(), meta_user_rlock(), meta_user_unlock(), meta_user_wlock(), NULLER, OBJECT_CONTACTS, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_CONTACTS_EDIT, PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), sess_serial_check(), st_char_get(), st_length_get(), and status.

5.456.3.22 void portal_endpoint_contacts_list (connection_t * con)

List a user's contacts in response to a "contacts.list" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2477 of file endpoint.c.

References `contact_detail_t`, `contact_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_object_set_new_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_string_d`, `json_unpack_ex_d`, `log_pedantic`, `magma_folder_find_number()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_LIST`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, and `st_length_get()`.

5.456.3.23 void portal_endpoint_contacts_load (connection_t * con)

Return information for a portal "contacts.load" json-rpc request.

Note:

This function returns the all the stored details about a specified contact entry.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2834 of file endpoint.c.

References `contact_t`, `inx_find()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `M_TYPE_UINT64`, `magma_folder_find_number()`, `portal_contact_details()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_LOAD`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

5.456.3.24 void portal_endpoint_contacts_move (connection_t * con)

Move a user's contact entry to another contacts folder in response to a "contacts.move" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2359 of file endpoint.c.

References `contact_find_number()`, `contact_move()`, `contact_t`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `magma_folder_find_number()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_MOVE`, `PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `sess_serial_check()`, `st_char_get()`, `st_length_get()`, and `status`.

5.456.3.25 void portal_endpoint_contacts_remove (connection_t * con)

Remove a user's contact entry in response to a "contacts.remove" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2422 of file endpoint.c.

References `contact_delete()`, `contact_find_number()`, `contact_t`, `inx_delete()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `M_TYPE_UINT64`, `magma_folder_find_number()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONTACTS_REMOVE`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `sess_serial_check()`, `st_char_get()`, `st_length_get()`, `status`, `multi_t::u64`, and `multi_t::val`.

5.456.3.26 void portal_endpoint_cookies (connection_t * con)

Return whether or not a user is using cookies, in response to a "cookies" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 564 of file endpoint.c.

References `PLACER`, `PORTAL_ENDPOINT_ERROR_COOKIES`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_cmp_ci_starts()`.

5.456.3.27 void portal_endpoint_error (connection_t * con, int_t http_code, int_t error_code, chr_t * message)

Return a json-rpc error response to the remote client.

Parameters:

con a pointer to the connection object across which the response will be sent.

http_code the http response code to be sent to the remote client in the response header.

error_code the numerical error code to be encoded in the json-rpc error message.

message a descriptive error string to be encoded in the json-rpc error message.

Returns:

This function returns no value.

Definition at line 124 of file endpoint.c.

References `con_write_st()`, `HTTP_ERROR_500`, `http_response_header()`, `json_decref_d`, `json_dumps_d`, `json_pack_ex_d`, `log_options`, `log_pedantic`, `M_LOG_CRITICAL`, `M_LOG_STACK_TRACE`, `magma`, `ns_append()`, `ns_free()`, `ns_length_get()`, `NULLER`, `PLACER`, `magma_t::portal`, and `magma_t::web`.

Referenced by `portal_endpoint()`, `portal_endpoint_ad()`, `portal_endpoint_alert_acknowledge()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_auth()`, `portal_endpoint_config_edit()`, `portal_endpoint_config_load()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_folders_add()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_remove()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_endpoint_scrape()`, `portal_endpoint_scrape_add()`, `portal_folder_contacts_add()`, `portal_folder_contacts_remove()`, `portal_folder_mail_add()`, `portal_folder_mail_remove()`, `portal_meta()`, `portal_settings_changepass()`, `portal_settings_identity()`, and `portal_validate_request()`.

5.456.3.28 void portal_endpoint_folders_add (connection_t * con)

Create a new folder in response to a "folders.add" json-rpc portal request.

Note:

Currently only contexts for new mail folders and contacts folders are supported.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 587 of file endpoint.c.

References `count`, `json_object_size_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `ns_length_get()`, `NULLER`, `PORTAL_ENDPOINT_CONTEXT_CONTACTS`, `PORTAL_ENDPOINT_CONTEXT_INVALID`, `PORTAL_ENDPOINT_CONTEXT_MAIL`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_FOLDERS_ADD`, `PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD`, `portal_folder_contacts_add()`, `portal_folder_mail_add()`, `portal_parse_context()`, `portal_validate_request()`, `st_char_get()`, `st_free()`, `st_import()`, and `st_length_get()`.

5.456.3.29 void portal_endpoint_folders_list (connection_t * con)

Return a list of a user's folders in response to a "folders.list" json-rpc portal request.

Note:

Currently only contexts for new mail folders and contacts folders are supported.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

LOW: User created folder names `_could_` have a NULL byte. How should we handle that?

LOW: User created folder names `_could_` have a NULL byte. How should we handle that?

Definition at line 649 of file endpoint.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_unpack_ex_d`, `log_pedantic`, `magma_folder_t`, `meta_folder_t`, `NULLER`, `PLACER`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_FOLDERS_LIST`, `PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, `st_cmp_ci_eq()`, and `st_length_get()`.

5.456.3.30 void portal_endpoint_folders_remove (connection_t * con)

Remove a user's folder in response to a json-rpc "folders.remove" portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 732 of file endpoint.c.

References JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, NULLER, PORTAL_ENDPOINT_CONTEXT_CONTACTS, PORTAL_ENDPOINT_CONTEXT_INVALID, PORTAL_ENDPOINT_CONTEXT_MAIL, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, portal_folder_contacts_remove(), portal_folder_mail_remove(), portal_parse_context(), portal_validate_request(), st_char_get(), and st_length_get().

5.456.3.31 void portal_endpoint_folders_rename (connection_t * con)

Rename a user's folder in response to a json-rpc "folders.rename" portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

HIGH: Use the context and act appropriately!

Definition at line 773 of file endpoint.c.

References contact_folder_rename(), imap_folder_rename(), JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, magma_folder_find_number(), magma_folder_name(), magma_folder_t, meta_folder_t, meta_folders_by_number(), meta_folders_name(), meta_user_unlock(), meta_user_wlock(), ns_length_get(), NULLER, OBJECT_CONTACTS, OBJECT_FOLDERS, PORTAL_ENDPOINT_CONTEXT_CONTACTS, PORTAL_ENDPOINT_CONTEXT_INVALID, PORTAL_ENDPOINT_CONTEXT_MAIL, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION, PORTAL_ENDPOINT_ERROR_FOLDERS_RENAME, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_parse_context(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_free(), st_import(), and st_length_get().

5.456.3.32 void portal_endpoint_folders_tags (connection_t * con)

HIGH: Right now we only have indexes with the mail folders so other context types generate an empty array result.

Definition at line 918 of file endpoint.c.

References meta_stats_tag_t::count, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_free(), inx_t, json_decref_d, json_integer_d, json_object_d, json_object_set_new_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, meta_folders_by_number(), meta_folders_stats_tags(), meta_user_rlock(), meta_user_unlock(), NULLER, PORTAL_ENDPOINT_CONTEXT_MAIL, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_FOLDERS_TAGS, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_parse_context(), portal_validate_request(), st_char_get(), st_length_get(), stats, and meta_stats_tag_t::tag.

5.456.3.33 void portal_endpoint_logout (connection_t * con)

Log a user out of a portal session in response to a "logout" json-rpc portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

Definition at line 374 of file endpoint.c.

References HTTP_COOKIE_DELETE, META_PROTOCOL_WEB, meta_user_ref_dec(), PORTAL_ENDPOINT_ERROR_LOGOUT, portal_endpoint_response(), portal_validate_request(), and SESSION_STATE_TERMINATED.

5.456.3.34 void portal_endpoint_messages_compose (connection_t * con)

Compose a new message in response to a "messages.compose" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 1942 of file endpoint.c.

References composition_t, inx_alloc(), inx_find(), inx_insert(), json_object_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, log_error, M_INX_LINKED, M_TYPE_UINT64, mm_alloc(), mutex_lock(), mutex_unlock(), portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_MESSAGES_COMPOSE, portal_endpoint_response(), portal_validate_request(), sess_release_attachment(), sess_release_composition(), multi_t::u64, and multi_t::val.

5.456.3.35 void portal_endpoint_messages_copy (connection_t * con)

HIGH: If a multiple message copy fails in the middle, go back and remove any messages that may have been copied before the error.

Definition at line 1114 of file endpoint.c.

References count, inx_find(), json_array_append_new_d, json_array_d, json_array_get_d, json_array_size_d, json_decref_d, json_integer_value_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_TYPE_UINT64, meta_folders_by_number(), META_LOCKED, meta_message_t, meta_messages_copier(), meta_messages_update_sequences(), meta_user_unlock(), meta_user_wlock(), OBJECT_MESSAGES, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_MESSAGES_COPY, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.456.3.36 void portal_endpoint_messages_flag (connection_t * con)

HIGH: This function needs restructuring pretty badly. We are setting collection to NULL so we can add a cleanup call below. But if the function works as intended the reference is stolen by the response. but if an error occurs and the we don't return a response the collection ends up as a memory leak without the cleanup call below.

TODO: Were holding onto the user lock while waiting on the response to be sent over the wire; and this function could probably use a rethink.

HIGH: See the sister comment a few lines back!

Definition at line 1203 of file endpoint.c.

References count, inx_alloc(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), inx_find(), inx_free(), inx_insert(), inx_t, json_array_append_new_d, json_array_d, json_array_get_d, json_array_size_d, json_decref_d, json_integer_value_d, json_object_size_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_INX_LINKED, M_TYPE_UINT64, MAIL_STATUS_SYSTEM_FLAGS,

MAIL_STATUS_USER_FLAGS, meta_data_flags_add(), meta_data_flags_remove(), meta_data_flags_replace(), meta_folders_by_number(), meta_message_t, meta_user_rlock(), meta_user_unlock(), meta_user_wlock(), NULLER, OBJECT_MESSAGES, PLACER, PORTAL_ENDPOINT_ACTION_ADD, PORTAL_ENDPOINT_ACTION_LIST, PORTAL_ENDPOINT_ACTION_REMOVE, PORTAL_ENDPOINT_ACTION_REPLACE, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, PORTAL_ENDPOINT_ERROR_MESSAGES_FLAG, PORTAL_ENDPOINT_ERROR_REFERENCE, PORTAL_ENDPOINT_ERROR_SYSTEM_FLAG, portal_endpoint_response(), portal_message_flags_array(), portal_parse_flags(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_cmp_ci_eq(), st_length_get(), multi_t::u64, and multi_t::val.

5.456.3.37 void portal_endpoint_messages_list (connection_t * con)

Retrieve a list of the user's messages in response to a json-rpc "messages.list" portal request.

Parameters:

con a pointer to the connection object of the requesting user.

Returns:

This function returns no value.

LOW: Add the ability to track the recipient email address for a message, even if its not provided in the To field.

LOW: Add snippet support.

Definition at line 991 of file endpoint.c.

References ar_field_st(), ar_length_get(), count, inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), json_array_append_new_d, json_array_d, json_decref_d, json_object_size_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_string_d, json_unpack_ex_d, log_pedantic, mail_header_fetch_cleaned(), mail_load_header(), meta_message_t, meta_user_rlock(), meta_user_unlock(), PLACER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_MESSAGES_LIST, portal_endpoint_response(), portal_message_flags_array(), portal_validate_request(), st_char_get(), st_cleanup, st_free(), st_import(), and st_length_get().

5.456.3.38 void portal_endpoint_messages_load (connection_t * con)

Definition at line 1836 of file endpoint.c.

References data, inx_find(), json_decref_d, json_object_d, json_object_set_new_d, json_object_size_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_TYPE_UINT64, mail_destroy(), mail_load_message(), mail_mime_update(), meta_message_t, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, PORTAL_ENDPOINT_ERROR_MESSAGES_LOAD, PORTAL_ENDPOINT_ERROR_READ, PORTAL_ENDPOINT_ERROR_REFERENCE, PORTAL_ENDPOINT_MESSAGE_ATTACHMENTS, PORTAL_ENDPOINT_MESSAGE_BODY, PORTAL_ENDPOINT_MESSAGE_HEADER, PORTAL_ENDPOINT_MESSAGE_INFO, PORTAL_ENDPOINT_MESSAGE_META, PORTAL_ENDPOINT_MESSAGE_SECURITY, PORTAL_ENDPOINT_MESSAGE_SERVER, PORTAL_ENDPOINT_MESSAGE_SOURCE, portal_endpoint_response(), portal_message_attachments(), portal_message_body(), portal_message_header(), portal_message_info(), portal_message_meta(), portal_message_security(), portal_message_server(), portal_message_source(), portal_parse_sections(), portal_validate_request(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.456.3.39 void portal_endpoint_messages_move (connection_t * con)

HIGH: If a multiple message copy fails in the middle, go back and remove any messages that may have been copied before the error.

Definition at line 1404 of file endpoint.c.

References count, inx_find(), json_array_get_d, json_array_size_d, json_integer_value_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_TYPE_UINT64, meta_folders_by_number(), META_LOCKED, meta_message_t, meta_messages_mover(), meta_messages_update_sequences(), meta_user_unlock(),

meta_user_wlock(), OBJECT_MESSAGES, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, PORTAL_ENDPOINT_ERROR_MESSAGES_MOVE, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.456.3.40 void portal_endpoint_messages_remove (connection_t * con)

Remove a user's mail message in response to a "messages.remove" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 1500 of file endpoint.c.

References count, inx_delete(), inx_find(), json_array_get_d, json_array_size_d, json_integer_value_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_pedantic, M_TYPE_UINT64, mail_remove_message(), meta_folders_by_number(), meta_message_t, meta_messages_update_sequences(), meta_user_unlock(), meta_user_wlock(), OBJECT_MESSAGES, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_MESSAGES_REMOVE, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_validate_request(), serial_get(), serial_increment(), sess_trigger(), st_char_get(), st_length_get(), multi_t::u64, and multi_t::val.

5.456.3.41 void portal_endpoint_messages_send (connection_t * con)

Send a composed message in response to a "messages.send" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 2136 of file endpoint.c.

References composition_t, inx_delete(), inx_find(), inx_free(), inx_t, json_object_size_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, json_unpack_ex_d, log_error, log_pedantic, M_TYPE_UINT64, mutex_lock(), mutex_unlock(), NULLER, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, PORTAL_ENDPOINT_ERROR_MESSAGES_SEND, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), portal_outbound_checks(), portal_parse_json_str_array(), portal_smtp_create_data(), portal_smtp_relay_message(), portal_validate_request(), st_char_get(), st_free(), st_length_get(), multi_t::u64, and multi_t::val.

5.456.3.42 void portal_endpoint_messages_tag (connection_t * con)

Perform a tag operation on a user's message, in response to a "messages.tag" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

LOW: We don't need to add the flag to every message. Just the ones that don't already have the flag.

LOW: Like the line, were not checking whether the message even needs to have the flag removed. Were also not handling remove requests that result in a message having no tags.

If this is going supposed to be a replace operation, we truncate all of the message tags so we can insert the replacements below.

TODO: This is ugly. Were rebuilding the tags array from the database on each iteration because its easier than trying to figure out which slot needs to be removed.

TODO: If the update is triggered before the tagged flag is added to the database the refresh might not pull the data for who recently got tagged! We need to need to look for this type of sequence bug elsewhere too.

LOW: Were holding onto the user lock while were waiting on the response to be sent over the wire; and this function could probably use a rethink.

Definition at line 1640 of file endpoint.c.

References `ar_field_st()`, `ar_free()`, `ar_length_get()`, `inx_alloc()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_find()`, `inx_free()`, `inx_insert()`, `inx_t`, `json_array_append_new_d`, `json_array_d`, `json_array_get_d`, `json_array_size_d`, `json_decref_d`, `json_integer_value_d`, `json_object_size_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS`, `json_string_d`, `json_string_value_d`, `json_unpack_ex_d`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_UINT64`, `MAIL_STATUS_TAGGED`, `meta_data_delete_tag()`, `meta_data_fetch_message_tags()`, `meta_data_flags_add()`, `meta_data_flags_remove()`, `meta_data_insert_tag()`, `meta_data_truncate_tags()`, `meta_folders_by_number()`, `meta_message_t`, `meta_user_rlock()`, `meta_user_serial_check()`, `meta_user_unlock()`, `meta_user_wlock()`, `NULLER`, `OBJECT_MESSAGES`, `PLACER`, `PORTAL_ENDPOINT_ACTION_ADD`, `PORTAL_ENDPOINT_ACTION_LIST`, `PORTAL_ENDPOINT_ACTION_REMOVE`, `PORTAL_ENDPOINT_ACTION_REPLACE`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION`, `PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD`, `PORTAL_ENDPOINT_ERROR_MESSAGES_TAG`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `portal_validate_request()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_length_get()`, `multi_t::u64`, and `multi_t::val`.

5.456.3.43 void portal_endpoint_messages_tags (connection_t * con)

Get all of the message tags used by a specified user in response to a "messages.tags" json-rpc portal request.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 1589 of file endpoint.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `inx_free()`, `inx_t`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `json_string_d`, `log_error`, `log_pedantic`, `meta_data_fetch_all_tags()`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_MESSAGES_TAGS`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_char_get()`.

5.456.3.44 void portal_endpoint_response (connection_t * con, chr_t * format, ...)

Generate a json-rpc 2.0 response to a portal request.

See also:

`json_vpack_ex()`

Note:

This function indents the json response if specified in the configuration, and also automatically decreases the reference count of any json object that was packed for the reply.

Parameters:

- con* a pointer to the connection object across which the portal response will be sent.
- format* a pointer to a format string specifying the construction of the json-rpc response.
- ... a variable arguments style list of parameters to be passed to the json packing function.

Returns:

This function returns no value.

Definition at line 162 of file endpoint.c.

References `con_write_st()`, `HTTP_ERROR_500`, `http_response_header()`, `json_decref_d`, `json_dumps_d`, `json_vpack_ex_d`, `log_pedantic`, `magma`, `ns_append()`, `ns_free()`, `ns_length_get()`, `NULLER`, `PLACER`, `magma_t::portal`, and `magma_t::web`.

Referenced by `portal_endpoint_ad()`, `portal_endpoint_alert_acknowledge()`, `portal_endpoint_alert_list()`, `portal_endpoint_aliases()`, `portal_endpoint_attachments_add()`, `portal_endpoint_attachments_remove()`, `portal_endpoint_auth()`, `portal_endpoint_config_edit()`, `portal_endpoint_config_load()`, `portal_endpoint_contacts_add()`, `portal_endpoint_contacts_copy()`, `portal_endpoint_contacts_edit()`, `portal_endpoint_contacts_list()`, `portal_endpoint_contacts_load()`, `portal_endpoint_contacts_move()`, `portal_endpoint_contacts_remove()`, `portal_endpoint_cookies()`, `portal_endpoint_folders_list()`, `portal_endpoint_folders_rename()`, `portal_endpoint_folders_tags()`, `portal_endpoint_logout()`, `portal_endpoint_messages_compose()`, `portal_endpoint_messages_copy()`, `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, `portal_endpoint_messages_load()`, `portal_endpoint_messages_move()`, `portal_endpoint_messages_remove()`, `portal_endpoint_messages_send()`, `portal_endpoint_messages_tag()`, `portal_endpoint_messages_tags()`, `portal_folder_contacts_add()`, `portal_folder_contacts_remove()`, `portal_folder_mail_add()`, `portal_folder_mail_remove()`, `portal_meta()`, `portal_settings_changepass()`, and `portal_settings_identity()`.

5.456.3.45 void portal_endpoint_scrape (connection_t * con)**Note:**

This function is not implemented and may be removed entirely in the future.

Parameters:

- con* a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5767 of file endpoint.c.

References `JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_AD`, and `portal_validate_request()`.

5.456.3.46 void portal_endpoint_scrape_add (connection_t * con)**Note:**

This function is not implemented and may be removed entirely in the future.

Parameters:

- con* a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5751 of file endpoint.c.

References `JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_AD`, and `portal_validate_request()`.

5.456.3.47 void portal_endpoint_search (connection_t * con)

Definition at line 5729 of file endpoint.c.

References con_write_st(), HTTP_ERROR_500, http_response_header(), PLACER, PORTAL_ENDPOINT_ERROR_SEARCH, st_aprint(), st_free(), and st_length_get().

5.456.3.48 void portal_endpoint_sort (void)

Sort the web portal JSON command table to be ready for binary searches. TODO: We use session locks to synchronize access to the user context. Should we also be using mailbox cluster locks? Cluster level locking might be appropriate for operations that delete data likes folders.remove and messages.remove.

Returns:

This function returns no value.

Definition at line 21 of file endpoint.c.

References command_t, portal_endpoint_compare(), and portal_methods.

Referenced by protocol_init().

5.456.3.49 void portal_folder_contacts_add (connection_t * con, stringer_t * name, uint64_t parent)

folders.c folders.c

Note:

If parent is 0, then the new folder is assumed to be a root folder. This function returns no value, but returns the appropriate portal json-rpc response directly.

Parameters:

con a pointer to the connection object across which the add folder response will be sent.

name a managed string containing the name of the new folder.

parent the numerical id of the folder that will be the parent of the new folder (or 0 to specify a root folder).

Returns:

This function returns no value.

Definition at line 82 of file folders.c.

References contact_folder_create(), JSON_RPC_2_ERROR_SERVER_INTERNAL, magma_folder_find_name(), magma_folder_find_number(), meta_user_unlock(), meta_user_wlock(), OBJECT_CONTACTS, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_FOLDERS_ADD, PORTAL_ENDPOINT_ERROR_REFERENCE, portal_endpoint_response(), sess_serial_check(), and st_cleanup.

Referenced by portal_endpoint_folders_add().

5.456.3.50 void portal_folder_contacts_remove (connection_t * con, uint64_t foldernum)

Remove a user's contacts folder.

Parameters:

con a pointer to the connection object across which the remove folder response will be sent.

foldernum the numerical id of the contacts folder that will be removed.

Returns:

This function returns no value.

Definition at line 201 of file folders.c.

References `contact_folder_remove()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `magma_folder_find_number()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_CONTACTS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION`, `PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, and `sess_serial_check()`.

Referenced by `portal_endpoint_folders_remove()`.

5.456.3.51 void portal_folder_mail_add (connection_t * con, stringer_t * name, uint64_t parent)

Add a new mail folder for a user.

Note:

If parent is 0, then the new folder is assumed to be a root folder. This function returns no value, but returns the appropriate portal json-rpc response directly.

Parameters:

con a pointer to the connection object across which the add folder response will be sent.

name a managed string containing the name of the new folder.

parent the numerical id of the folder that will be the parent of the new folder (or 0 to specify a root folder).

Returns:

This function returns no value.

Definition at line 20 of file folders.c.

References `imap_folder_create()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `meta_folder_t`, `meta_folders_by_name()`, `meta_folders_by_number()`, `meta_folders_name()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_FOLDERS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_FOLDERS_ADD`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `sess_serial_check()`, `st_cleanup`, and `st_merge`.

Referenced by `portal_endpoint_folders_add()`.

5.456.3.52 void portal_folder_mail_remove (connection_t * con, uint64_t foldernum)

Remove a user's mail folder.

Parameters:

con a pointer to the connection object across which the remove folder response will be sent.

foldernum the numerical id of the mail folder that will be removed.

Returns:

This function returns no value.

Definition at line 148 of file folders.c.

References `imap_folder_remove()`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `meta_folder_t`, `meta_folders_by_number()`, `meta_folders_name()`, `meta_user_unlock()`, `meta_user_wlock()`, `OBJECT_FOLDERS`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION`, `PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE`, `PORTAL_ENDPOINT_ERROR_REFERENCE`, `portal_endpoint_response()`, `sess_serial_check()`, and `st_cleanup`.

Referenced by `portal_endpoint_folders_remove()`.

5.456.3.53 `json_t* portal_message_attachments (meta_message_t * meta, mail_message_t * data)`

messages.c

LOW: We also need to detect messages that have no readable content so the entire body is just a blob.

Create a function to search for the filename. Common locations are: Content-Type: image/png; name="webmail-php-download.png" Content-Disposition: attachment; filename="webmail-php-download.png"

Definition at line 164 of file messages.c.

References `ar_field_ptr()`, `ar_length_get()`, `mail_mime_t::body`, `mail_mime_t::children`, `count`, `mail_mime_t::header`, `json_array_append_new_d`, `json_array_d`, `json_decref_d`, `json_pack_ex_d`, `log_pedantic`, `mail_mime_type_group()`, `mail_mime_type_sub()`, `MESSAGE_TYPE_HTML`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_PLAIN`, `mail_message_t::mime`, `PLACER`, `st_aprint()`, `st_char_get()`, `st_cleanup`, `st_length_get()`, `st_merge`, `mail_mime_t::type`, and `type()`.

Referenced by `portal_endpoint_messages_load()`.

5.456.3.54 `json_t* portal_message_body (meta_message_t * meta, mail_message_t * data)`

TODO: I consider it a miracle whenever the above logic actually manages to select the message content. But if it doesn't we fall through to this fixed error message, at least until we can improve the selection process.

HIGH: Because JSON wants NULLERS, and were using PLACEHOLDERS, its printing out the entire message, not just the section of interest.

Definition at line 108 of file messages.c.

References `ar_field_ptr()`, `mail_mime_t::body`, `mail_mime_t::children`, `CONTIGUOUS`, `HEAP`, `json_pack_ex_d`, `log_pedantic`, `MAIL_MIME_RECURSION_LIMIT`, `MESSAGE_TYPE_HTML`, `MESSAGE_TYPE_MULTI_ALTERNATIVE`, `MESSAGE_TYPE_MULTI_MIXED`, `MESSAGE_TYPE_MULTI_RELATED`, `MESSAGE_TYPE_MULTI_UNKOWN`, `MESSAGE_TYPE_PLAIN`, `mail_message_t::mime`, `NULLER_T`, `pl_char_get()`, `pl_length_get()`, `PLACER`, `st_char_get()`, `st_cleanup`, `st_dupe_opts()`, `st_nullify()`, `mail_mime_t::type`, and `type()`.

Referenced by `portal_endpoint_messages_load()`.

5.456.3.55 `json_t* portal_message_flags_array (meta_message_t * meta)`

flags.c flags.c

TODO: The messages.load method uses the flags/tags helper functions, but the messages.list and the messages.tags/flags methods still need to be updated.

Parameters:

meta a pointer to the meta message object of the mail message to have its flags parsed.

Returns:

NULL on failure, or a pointer to the json object of the specified mail message's flags.

Definition at line 18 of file flags.c.

References `json_array_append_new_d`, `json_array_d`, `json_string_d`, `MAIL_MARK_BLACKHOLED`, `MAIL_MARK_INFECTED`, `MAIL_MARK_JUNK`, `MAIL_MARK_PHISHING`, `MAIL_MARK_SPOOFED`, `MAIL_STATUS_ANSWERED`, `MAIL_STATUS_APPENDED`, `MAIL_STATUS_DELETED`, `MAIL_STATUS_DRAFT`, `MAIL_STATUS_EMPTY`, `MAIL_STATUS_ENCRYPTED`, `MAIL_STATUS_FLAGGED`, `MAIL_STATUS_RECENT`, and `MAIL_STATUS_SEEN`.

Referenced by `portal_endpoint_messages_flag()`, `portal_endpoint_messages_list()`, and `portal_message_meta()`.

5.456.3.56 `json_t* portal_message_header (meta_message_t * meta, mail_message_t * data)`

Build the "header" section response to a rpc-json "messages.load" request.

Note:

The following header fields will be returned: To, CC, BCC, From, Replyto, Sender, Return-Path, Subject, Date, and Size.

Parameters:

meta a pointer to the meta message object of the requested message.

data a pointer to the mail message object containing the requested message's data.

Returns:

a pointer to a json object containing the header fields of the requested message.

Definition at line 79 of file messages.c.

References mail_message_t::header_length, json_pack_ex_d, log_pedantic, mail_header_fetch_cleaned(), mm_wipe(), PLACER, st_char_get(), st_cleanup, and mail_message_t::text.

Referenced by portal_endpoint_messages_load().

5.456.3.57 json_t* portal_message_info (meta_message_t * meta)

Build the "info" section response to a rpc-json "messages.load" request.

Parameters:

meta a pointer to the meta message object of the requested message.

Returns:

a pointer to a json object containing the appropriate information about the requested message.

Definition at line 270 of file messages.c.

References json_pack_ex_d, and log_pedantic.

Referenced by portal_endpoint_messages_load().

5.456.3.58 json_t* portal_message_meta (meta_message_t * meta)

Definition at line 13 of file messages.c.

References json_pack_ex_d, log_pedantic, portal_message_flags_array(), and portal_message_tags_array().

Referenced by portal_endpoint_messages_load().

5.456.3.59 json_t* portal_message_security (meta_message_t * meta)

TODO: Replace hard coded values with actual data.

Definition at line 42 of file messages.c.

References json_pack_ex_d, and log_pedantic.

Referenced by portal_endpoint_messages_load().

5.456.3.60 json_t* portal_message_server (meta_message_t * meta)

TODO: Replace hard coded values with actual data.

Definition at line 56 of file messages.c.

References json_pack_ex_d, and log_pedantic.

Referenced by portal_endpoint_messages_load().

5.456.3.61 json_t* portal_message_source (meta_message_t * meta)

TODO: Replace hard coded values with actual data.

Definition at line 27 of file messages.c.

References json_pack_ex_d, log_pedantic, and rand_get_int64().

Referenced by portal_endpoint_messages_load().

5.456.3.62 json_t* portal_message_tags_array (meta_message_t * meta)

Definition at line 86 of file flags.c.

References ar_field_st(), ar_length_get(), count, json_array_append_new_d, json_array_d, json_string_d, and st_char_get().

Referenced by portal_message_meta().

5.456.3.63 void portal_meta (connection_t * con)

Return information for a portal "meta" json-rpc request.

Note:

This function returns various pieces of information about the user such as their plan type, quota, etc.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5834 of file endpoint.c.

References con_addr_presentation(), json_object_d, json_pack_ex_d, JSON_RPC_2_ERROR_SERVER_INTERNAL, log_pedantic, MAGMA_PORTAL_VERSION, MANAGEDBUF, portal_endpoint_error(), PORTAL_ENDPOINT_ERROR_META, portal_endpoint_response(), portal_validate_request(), and st_char_get().

5.456.3.64 bool_t portal_outbound_checks (uint64_t usernum, stringer_t * username, stringer_t * verification, stringer_t * from, size_t num_recipients, stringer_t * body_plain, stringer_t * body_html, chr_t ** errmsg)

[mail.c](#) [mail.c](#)

See also:

[smtp_data_outbound\(\)](#)

Note:

The checks include: Pattern matching for junk mail, authorization for the user to use the email address or domain specified in the From address, and finally, a virus check.

Parameters:

cred a credential structure obtained for the authenticated user's session.

usernum the numerical id of the user attempting to send the message.

from a managed string containing the email address specified as the From address.

nrecipients the number of recipients that will receive the sent message.

body_plain a managed string containing the plain text body of the message.

body_html a managed string containing the html body of the message.

errmsg the address of a pointer to a null-terminated string that will be set to a descriptive error message on failure.

Returns:

true if all security checks were passed or false otherwise.

Definition at line 24 of file mail.c.

References pattern_check(), smtp_check_authorized_from(), smtp_check_transmit_quota(), smtp_fetch_authorization(), and virus_check().

Referenced by portal_endpoint_messages_send().

5.456.3.65 int_t portal_parse_action (stringer_t * action)

Note:

This function is currently not called anywhere in the code and may be subject to removal.

Definition at line 123 of file parse.c.

References PLACER, PORTAL_ENDPOINT_ACTION_ADD, PORTAL_ENDPOINT_ACTION_INVALID, PORTAL_ENDPOINT_ACTION_LIST, PORTAL_ENDPOINT_ACTION_REMOVE, PORTAL_ENDPOINT_ACTION_REPLACE, and st_cmp_ci_eq().

5.456.3.66 int_t portal_parse_context (stringer_t * context)

Parse the context of a requested folder.

Note:

If no context is specified, the "mail" context is assumed (PORTAL_ENDPOINT_CONTEXT_MAIL).

Parameters:

context a managed string containing the context (supported values are "mail", "contacts", "settings", and "help").

Returns:

PORTAL_ENDPOINT_CONTEXT_INVALID on failure, or the flag of the determined context on success.

Definition at line 92 of file parse.c.

References PLACER, PORTAL_ENDPOINT_CONTEXT_CONTACTS, PORTAL_ENDPOINT_CONTEXT_HELP, PORTAL_ENDPOINT_CONTEXT_INVALID, PORTAL_ENDPOINT_CONTEXT_MAIL, PORTAL_ENDPOINT_CONTEXT_SETTINGS, and st_cmp_ci_eq().

Referenced by portal_endpoint_folders_add(), portal_endpoint_folders_remove(), portal_endpoint_folders_rename(), and portal_endpoint_folders_tags().

5.456.3.67 int_t portal_parse_flags (json_t * array, uint32_t * flags)

Definition at line 46 of file flags.c.

References count, json_array_get_d, json_array_size_d, json_string_value_d, MAIL_MARK_BLACKHOLED, MAIL_MARK_INFECTED, MAIL_MARK_JUNK, MAIL_MARK_PHISHING, MAIL_MARK_SPOOFED, MAIL_STATUS_ANSWERED, MAIL_STATUS_APPENDED, MAIL_STATUS_DELETED, MAIL_STATUS_DRAFT, MAIL_STATUS_FLAGGED, MAIL_STATUS_RECENT, MAIL_STATUS_SEEN, NULLER, PLACER, and st_cmp_ci_eq().

Referenced by portal_endpoint_messages_flag().

5.456.3.68 `inx_t* portal_parse_json_str_array (json_t * json, size_t * nout)`

parse.c parse.c

Parameters:

json a json object containing an array of strings.

nout if not NULL, an optional pointer to a size_t to receive the item count of the json string array.

Returns:

NULL on failure, or a pointer to an inx holder containing the specified json array contents as a collection of managed strings.

Definition at line 16 of file parse.c.

References count, inx_alloc(), inx_free(), inx_insert(), inx_t, json_array_get_d, json_array_size_d, json_string_value_d, log_error, log_pedantic, M_INX_LINKED, M_TYPE_UINT64, ns_length_get(), st_free(), st_import(), multi_t::u64, and multi_t::val.

Referenced by portal_endpoint_messages_send().

5.456.3.69 `int_t portal_parse_sections (json_t * array, uint32_t * sections)`

Parse the json-rpc messages.load parameter "section" from an array of strings into a bitmask of section flags.

Parameters:

array a pointer to the json object representing the "section" string array of the json request message.

sections a pointer to an unsigned 32 bit integer that will receive the translated section flags on success.

Returns:

< 0 on error (-3 for an internal server error, -2 if an unknown flag was received, or -1 on syntax error), 0 for an empty sections array, or 1 on success.

Definition at line 159 of file parse.c.

References count, json_array_get_d, json_array_size_d, json_string_value_d, NULLER, PLACER, PORTAL_ENDPOINT_MESSAGE_ATTACHMENTS, PORTAL_ENDPOINT_MESSAGE_BODY, PORTAL_ENDPOINT_MESSAGE_HEADER, PORTAL_ENDPOINT_MESSAGE_INFO, PORTAL_ENDPOINT_MESSAGE_META, PORTAL_ENDPOINT_MESSAGE_SECURITY, PORTAL_ENDPOINT_MESSAGE_SERVER, PORTAL_ENDPOINT_MESSAGE_SOURCE, and st_cmp_ci_eq().

Referenced by portal_endpoint_messages_load().

5.456.3.70 `void portal_print_login (connection_t * con, chr_t * message)`

[portal.c](#) [portal.c](#)

Note:

The portal login page can be found in 'portal/login.template'

Parameters:

con a pointer to the connection object of the client requesting the portal login page.

message a pointer to a null-terminated string that will be displayed to the user in the login page as the contents of the node with the id of "message" in the portal login template.

Returns:

This function returns no value.

Definition at line 18 of file portal.c.

References `con_write_st()`, `http_page_t::content`, `http_page_t::doc_obj`, `http_page_free()`, `http_page_get()`, `http_print_500()`, `http_response_header()`, `st_free()`, `st_length_get()`, `http_content_t::type`, `xml_dump_doc()`, `xml_node_set_content()`, `xml_xpath_eval()`, and `http_page_t::xpath_ctx`.

Referenced by `portal_process()`.

5.456.3.71 void portal_process (connection_t * con)

Process a connection to the web portal.

Note:

If `magma.web.portal.safeguard` is set, the user will be redirected to a secure login.

Parameters:

con a pointer to the connection of the http client requesting the portal.

Returns:

This function returns no value.

Definition at line 57 of file portal.c.

References `con_localhost()`, `con_secure()`, `http_parse_context()`, `http_print_301()`, `magma`, `PLACER`, `magma_t::portal`, `portal_print_login()`, and `magma_t::web`.

Referenced by `http_response()`.

5.456.3.72 void portal_settings_identity (connection_t * con)

Return information for a portal "settings.identity" json-rpc request.

Note:

This function returns the full name, first name, last name, and website of the requested user.

Parameters:

con a pointer to the connection object across which the json-rpc response will be sent.

Returns:

This function returns no value.

Definition at line 5801 of file endpoint.c.

References `json_object_d`, `json_pack_ex_d`, `JSON_RPC_2_ERROR_SERVER_INTERNAL`, `log_pedantic`, `portal_endpoint_error()`, `PORTAL_ENDPOINT_ERROR_SETTINGS_IDENTITY`, `portal_endpoint_response()`, `portal_validate_request()`, and `st_char_get()`.

5.456.3.73 stringer_t* portal_smtp_create_data (inx_t * attachments, stringer_t * from, inx_t * to, inx_t * cc, inx_t * bcc, stringer_t * subject, stringer_t * body_plain, stringer_t * body_html)

Create the data of an outbound smtp message that will be specified with the smtp DATA command.

Parameters:

attachments an optional inx holder containing a list of attachments to be included in the message.

from a managed string containing the email address sending the message.
to an inx holder containing a list of To: email recipients as managed strings.
cc an inx holder containing a list of 0 or more CC: recipients as managed strings.
bcc an inx holder containing a list of 0 or more BCC: recipients as managed strings.
subject a managed string containing the subject of the message.
body_plain an optional managed string containing the plain text body of the message.
body_html an optional managed string containing the html-formatted body of the message.

Returns:

NULL on failure or a managed string containing the packaged outbound smtp message data on success.

Definition at line 177 of file mail.c.

References `ar_append()`, `ar_free()`, `ar_length_get()`, `ARRAY_TYPE_POINTER`, `attachment_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_reset()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_pedantic`, `mail_mime_encode_part()`, `mail_mime_generate_boundary()`, `mail_mime_get_smtp_envelope()`, `st_free()`, and `st_merge`.

Referenced by `portal_endpoint_messages_send()`.

5.456.3.74 `stringer_t* portal_smtp_merge_headers (inx_t * headers, stringer_t * leading, stringer_t * trailing)`

Merge a list of smtp message headers into a single string, preceded by the leading text and followed by the trailing text.

Parameters:

headers an inx holder containing a collection of header string data to be merged together.
leading a managed string containing text that will lead each header line.
trailing a managed string containing text that will trail each header line.

Returns:

NULL on failure or a managed string containing the merged headers on success.

Definition at line 301 of file mail.c.

References `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `st_alloc()`, `st_cleanup`, and `st_merge`.

Referenced by `mail_mime_get_smtp_envelope()`.

5.456.3.75 `bool_t portal_smtp_relay_message (stringer_t * from, inx_t * to, stringer_t * data, size_t send_size, chr_t ** errmsg)`

Send (relay) a message composed by a user via a portal session.

See also:

[smtp_relay_message\(\)](#) - a lot of logic borrowed from here.

Parameters:

from a pointer to a managed string containing the email address specified as the From address.
to a pointer to a managed string containing the destination email address of the message.
data a pointer to a managed string containing the raw data of the mail message.
send_size if greater than 0, specify the optional SIZE parameter to the MAIL FROM command.
errmsg the address of a pointer to a null-terminated string that will be set to a descriptive error message on failure.

Returns:

true if the mail message was sent successfully, or false otherwise.

Definition at line 86 of file mail.c.

References `client_t`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `smtp_client_close()`, `smtp_client_connect()`, `smtp_client_send_data()`, `smtp_client_send_helo()`, `smtp_client_send_mailfrom()`, and `smtp_client_send_rcptto()`.

Referenced by `portal_endpoint_messages_send()`.

5.456.3.76 void portal_upload (connection_t * con)

Process uploaded attachments for messages composed in conjunction with the portal interface.

Definition at line 6074 of file endpoint.c.

References `attachment_t`, `con_localhost()`, `con_secure()`, `data`, `get_header_opt()`, `http_data_free()`, `http_data_header_parse_line()`, `HTTP_ERROR_401`, `HTTP_ERROR_403`, `HTTP_ERROR_404`, `HTTP_ERROR_405`, `HTTP_ERROR_500`, `HTTP_MERGED`, `HTTP_METHOD_POST`, `http_parse_context()`, `HTTP_PORTAL`, `http_print_301()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `log_error`, `log_pedantic`, `magma`, `multipart_get_boundary()`, `http_data_t::name`, `NULLER`, `pl_char_get()`, `pl_empty()`, `pl_length_get()`, `pl_shrink_before_characters()`, `pl_skip_characters()`, `pl_skip_to_characters()`, `PLACER`, `placer_t`, `magma_t::portal`, `portal_get_upload_attachment()`, `st_char_get()`, `st_cmp_ci_eq()`, `st_import()`, `st_length_get()`, `str_tok_get_bl()`, `str_tok_get_count_bl()`, `http_data_t::value`, and `magma_t::web`.

Referenced by `http_response()`.

5.457 src/web/register/captcha.c File Reference

```
#include "magma.h"
```

Functions

- [stringer_t * register_captcha_random_font](#) (void)

LOW: We shouldn't have to actually scan the fonts directory to find a valid file. Instead we could cache a list of valid fonts and then pick from it randomly.

- void [register_captcha_write_noise](#) (gdImagePtr image, [int_t](#) x, [int_t](#) y)

Fill an image's background partially with pixelated noise to make it more difficult to read.

- [stringer_t * register_captcha_generate](#) ([stringer_t](#) *value)

Generate a captcha image for a specified character string.

5.457.1 Function Documentation

5.457.1.1 [stringer_t * register_captcha_generate](#) ([stringer_t](#) * value)

Generate a captcha image for a specified character string. [captcha.c](#)

Parameters:

value a managed string containing the text that is to become the basis of the captcha challenge.

Returns:

NULL on failure, or a managed string containing the path to the image file containing the captcha graphic on success.

Definition at line 106 of file [captcha.c](#).

References [gdFree_d](#), [gdImageColorResolve_d](#), [gdImageCreate_d](#), [gdImageDestroy_d](#), [gdImageGifPtr_d](#), [gdImageStringFT_d](#), [log_pedantic](#), [mm_wipe\(\)](#), [rand_get_uint32\(\)](#), [register_captcha_random_font\(\)](#), [register_captcha_write_noise\(\)](#), [st_char_get\(\)](#), [st_import\(\)](#), and [st_length_get\(\)](#).

Referenced by [register_print_captcha\(\)](#).

5.457.1.2 [stringer_t * register_captcha_random_font](#) (void)

LOW: We shouldn't have to actually scan the fonts directory to find a valid file. Instead we could cache a list of valid fonts and then pick from it randomly. Select a random truetype font from the directory specified in [magma.http.fonts](#).

Returns:

NULL on failure, or a managed string containing the pathname of the randomly selected font file on success.

Definition at line 15 of file [captcha.c](#).

References [count](#), [magma_t::http](#), [log_pedantic](#), [magma](#), [NULLER](#), [PLACER](#), [rand_get_uint32\(\)](#), [st_aprint\(\)](#), and [st_cmp_ci_ends\(\)](#).

Referenced by [register_captcha_generate\(\)](#).

5.457.1.3 void register_captcha_write_noise (gdImagePtr *image*, int_t *x*, int_t *y*)

Fill an image's background partially with pixelated noise to make it more difficult to read.

Parameters:

- image* a pointer to the gd image to be modified.
- x* the height, in pixels, of the image region to be filled.
- y* the width, in pixels, of the image region to be filled.

Returns:

This function returns no value.

Definition at line 77 of file captcha.c.

References gdImageColorResolve_d, gdImageSetPixel_d, and rand_get_uint32().

Referenced by register_captcha_generate().

5.458 src/web/register/register.c File Reference

```
#include "magma.h"
```

Functions

- void [register_print_message](#) ([connection_t](#) *con, [chr_t](#) *message)
Display a custom message to the remote client using the register/message template.
- void [register_print_captcha](#) ([connection_t](#) *con, [register_session_t](#) *reg)
Print out a captcha challenge for a specified registration process.
- void [register_print_step1](#) ([connection_t](#) *con, [register_session_t](#) *reg, [chr_t](#) *message)
Display step 1 of the registration process (collect username, password, and captcha challenge).
- void [register_print_step2](#) ([connection_t](#) *con, [register_session_t](#) *reg, [chr_t](#) *message)
Display step 2 of the registration process.
- void [register_print_step3](#) ([connection_t](#) *con)
Display the registration step 3 template for a connection that has successfully created a new user.
- void [register_process](#) ([connection_t](#) *con)
Process a user registration request (/register).

5.458.1 Function Documentation

5.458.1.1 void [register_print_captcha](#) ([connection_t](#) * con, [register_session_t](#) * reg)

Print out a captcha challenge for a specified registration process. [register.c](#)

Parameters:

- con** the underlying client connection.
- reg** the pending registration session of the remote client.

Returns:

This function returns no value.

Definition at line 46 of file register.c.

References [con_print\(\)](#), [con_write_st\(\)](#), [http_print_500\(\)](#), [register_session_t::hvf_value](#), [log_info](#), [rand_choices\(\)](#), [register_captcha_generate\(\)](#), [st_char_get\(\)](#), [st_free\(\)](#), and [st_length_get\(\)](#).

Referenced by [register_process\(\)](#).

5.458.1.2 void [register_print_message](#) ([connection_t](#) * con, [chr_t](#) * message)

Display a custom message to the remote client using the register/message template.

Parameters:

- con** the client connection to receive the message.
- message** a pointer to a null-terminated string containing the custom message to be displayed inside the template sent to the remote client.

Returns:

This function returns no value.

Definition at line 17 of file register.c.

References `con_write_st()`, `content`, `http_get_template()`, `http_print_500()`, `http_response_header()`, `NULLER`, `PLACER`, `http_content_t::resource`, `st_dupes()`, `st_free()`, `st_length_get()`, `st_replace()`, and `http_content_t::type`.

Referenced by `register_abuse_checks()`.

5.458.1.3 void register_print_step1 (connection_t * con, register_session_t * reg, chr_t * message)

Display step 1 of the registration process (collect username, password, and captcha challenge).

Parameters:

con a pointer to the connection object of the remote client making the registration request.

reg a pointer to the pending registration session of the remote client.

message if not NULL, an optional error message to be displayed inside the registration step 1 template.

Returns:

This function returns no value.

Definition at line 78 of file register.c.

References `con_write_st()`, `content`, `http_get_template()`, `http_print_500_log()`, `http_response_header()`, `register_session_t::name`, `ns_length_get()`, `PLACER`, `http_content_t::resource`, `st_dupes()`, `st_free()`, `st_length_get()`, `st_replace()`, `http_content_t::type`, and `register_session_t::username`.

Referenced by `register_process()`.

5.458.1.4 void register_print_step2 (connection_t * con, register_session_t * reg, chr_t * message)

Display step 2 of the registration process.

Note:

Previously, this step used to obtain information like the desired user plan and billing info. At the moment, however, this step effectively does nothing but display a page to be clicked-through.

Parameters:

con a pointer to the connection object of the remote client making the registration request.

reg a pointer to the pending registration session of the remote client.

message if not NULL, an optional error message to be displayed inside the registration step 2 template.

Returns:

This function returns no value.

Definition at line 125 of file register.c.

References `con_write_st()`, `content`, `http_get_template()`, `http_print_500_log()`, `http_response_header()`, `register_session_t::name`, `ns_length_get()`, `PLACER`, `http_content_t::resource`, `st_dupes()`, `st_free()`, `st_length_get()`, `st_replace()`, and `http_content_t::type`.

Referenced by `register_process()`.

5.458.1.5 void register_print_step3 (connection_t * con)

Display the registration step 3 template for a connection that has successfully created a new user.

Parameters:

the remote connection making the registration request.

Returns:

This function returns no value.

Definition at line 164 of file register.c.

References con_write_st(), content, http_get_template(), http_print_500(), http_response_header(), http_content_t::resource, st_dupe(), st_free(), st_length_get(), and http_content_t::type.

Referenced by register_process().

5.458.1.6 void register_process (connection_t * con)

Process a user registration request (/register).

Note:

All requests will be redirected to a secure location. All new registration requests will also need to pass a test to ensure that they haven't been the product of abusive behavior. If an existing registration session can't be located, a new one will be generated. Depending on the user supplied http POST data, one of the three registration steps will be shown, or a captcha challenge will be presented to the user.

Returns:

This function returns no value.

Definition at line 190 of file register.c.

References con_secure(), HTTP_COMPLETE, HTTP_DATA_GET, http_data_get(), HTTP_DATA_POST, HTTP_ERROR_500, http_print_301(), log_pedantic, register_abuse_checks(), register_business_step1(), register_business_step2(), register_print_captcha(), register_print_step1(), register_print_step2(), register_print_step3(), register_session_cache(), register_session_free(), register_session_generate(), register_session_get(), register_session_t::usernum, and http_data_t::value.

Referenced by http_response().

5.459 src/web/register/register.h File Reference

Data Structures

- struct [register_session_t](#)

Defines

- #define [REGISTER_PASSWORD_MIN_LENGTH](#) 5
- #define [REGISTER_PASSWORD_MAX_LENGTH](#) 200
- #define [REGISTER_USERNAME_MIN_LENGTH](#) 1
- #define [REGISTER_USERNAME_MAX_LENGTH](#) 200

Functions

- [bool_t register_data_check_username](#) ([stringer_t](#) *username)
datatier.c
- [inx_t * register_data_fetch_blocklist](#) (void)
Fetch the blocklist for new user registration from the database.
- [int_t register_data_insert_user](#) ([connection_t](#) *con, [uint16_t](#) plan, [stringer_t](#) *username, [stringer_t](#) *password, [int64_t](#) transaction, [uint64_t](#) *outuser)
Insert a newly registered user into the database using information gathered by registration step #2.
- [bool_t register_abuse_check_blocklist](#) ([connection_t](#) *con)
abuse.c
- [bool_t register_abuse_checks](#) ([connection_t](#) *con)
Check to see if a registration request is allowed by a remote host; if not, display a banner.
- void [register_abuse_increment_history](#) ([connection_t](#) *con)
Increment the registration abuse counter for the requesting IP address.
- void [register_blocklist_free](#) (void)
Free a registration blocked list.
- void [register_blocklist_update](#) (void)
Update the registration blocklist from the database.
- [stringer_t * register_captcha_generate](#) ([stringer_t](#) *value)
captcha.c
- [stringer_t * register_captcha_random_font](#) (void)
LOW: We shouldn't have to actually scan the fonts directory to find a valid file. Instead we could cache a list of valid fonts and then pick from it randomly.
- void [register_captcha_write_noise](#) ([gdImagePtr](#) image, [int_t](#) x, [int_t](#) y)
Fill an image's background partially with pixelated noise to make it more difficult to read.
- [bool_t register_session_cache](#) ([connection_t](#) *con, [register_session_t](#) *session)
sessions.c

- void `register_session_free` (`register_session_t` *session)
Destroy a registration session and all its associated data.
- `register_session_t` * `register_session_generate` (void)
Generate a new registration session.
- `register_session_t` * `register_session_get` (`connection_t` *con, `stringer_t` *name)
Retrieve a registration session by a data key supplied in the user's http POST request.
- `chr_t` * `register_business_step1` (`connection_t` *con, `register_session_t` *reg)
business.c
- `chr_t` * `register_business_step2` (`connection_t` *con, `register_session_t` *reg)
Perform verification checking on all step 2 completed user fields, and display a welcome banner on success.
- `bool_t` `register_business_validate_password` (`stringer_t` *password)
Determine whether a registered password is valid.
- `int_t` `register_business_validate_username` (`stringer_t` *username)
Determine whether a registered username is valid.
- void `register_print_captcha` (`connection_t` *con, `register_session_t` *reg)
register.c
- void `register_print_message` (`connection_t` *con, `chr_t` *message)
Display a custom message to the remote client using the register/message template.
- void `register_print_step1` (`connection_t` *con, `register_session_t` *reg, `chr_t` *message)
Display step 1 of the registration process (collect username, password, and captcha challenge).
- void `register_print_step2` (`connection_t` *con, `register_session_t` *reg, `chr_t` *message)
Display step 2 of the registration process.
- void `register_print_step3` (`connection_t` *con)
Display the registration step 3 template for a connection that has successfully created a new user.
- void `register_process` (`connection_t` *con)
Process a user registration request (/register).

5.459.1 Define Documentation

5.459.1.1 #define REGISTER_PASSWORD_MAX_LENGTH 200

Definition at line 12 of file register.h.

Referenced by `register_business_step1()`, and `register_business_validate_password()`.

5.459.1.2 #define REGISTER_PASSWORD_MIN_LENGTH 5

Definition at line 11 of file register.h.

5.459.1.3 `#define REGISTER_USERNAME_MAX_LENGTH 200`

Definition at line 14 of file register.h.

Referenced by register_business_validate_username().

5.459.1.4 `#define REGISTER_USERNAME_MIN_LENGTH 1`

Definition at line 13 of file register.h.

5.459.2 Function Documentation

5.459.2.1 `bool_t register_abuse_check_blocklist (connection_t * con)`

abuse.c abuse.c

Parameters:

con the client connection to be checked.

Returns:

false if the check was passed, or true if the connection was made from an IP on the blocklist.

Definition at line 58 of file abuse.c.

References con_addr_standard(), inx_cursor_alloc(), inx_cursor_free(), inx_cursor_t, inx_cursor_value_next(), MANAGEDBUF, register_blocklist, register_blocklist_lock, rwlock_lock_read(), rwlock_unlock(), st_char_get(), st_cmp_ci_starts(), st_length_get(), and stats_increment_by_name().

Referenced by register_abuse_checks().

5.459.2.2 `bool_t register_abuse_checks (connection_t * con)`

Check to see if a registration request is allowed by a remote host; if not, display a banner.

Parameters:

con the remote connection to be checked.

Returns:

false if registration is allowed or true if not (remote host is on blocklist or registration has been throttled).

Definition at line 112 of file abuse.c.

References cache_get_u64(), con_addr_presentation(), con_addr_standard(), MANAGEDBUF, NULLER, register_abuse_check_blocklist(), register_print_message(), and st_char_get().

Referenced by register_process().

5.459.2.3 `void register_abuse_increment_history (connection_t * con)`

Increment the registration abuse counter for the requesting IP address.

Parameters:

con a pointer to the connection object of the client making the registration request.

Returns:

This function returns no value.

Definition at line 95 of file abuse.c.

References cache_increment(), con_addr_standard(), MANAGEDBUF, NULLER, and st_char_get().

Referenced by api_endpoint_register(), and register_business_step2().

5.459.2.4 void register_blocklist_free (void)

Free a registration blocked list.

Returns:

This function returns no value.

Definition at line 18 of file abuse.c.

References inx_free(), and register_blocklist.

5.459.2.5 void register_blocklist_update (void)

Update the registration blocklist from the database.

Returns:

This function returns no value.

Definition at line 32 of file abuse.c.

References inx_free(), inx_t, register_blocklist, register_blocklist_lock, register_data_fetch_blocklist(), rwlock_lock_write(), and rwlock_unlock().

5.459.2.6 chr_t* register_business_step1 (connection_t * con, register_session_t * reg)

business.c business.c

Note:

Checks include captcha verification, username validation, and password reentry verification and validation.

Parameters:

con the underlying client connection.

reg the underlying registration session.

Returns:

NULL on success, or a descriptive error string on failure.

Definition at line 107 of file business.c.

References data, http_data_get(), HTTP_DATA_POST, register_session_t::hvf_input, register_session_t::hvf_value, log_pedantic, lower_st(), magma, magma_t::minimum_password_length, register_session_t::password, register_business_validate_password(), register_business_validate_username(), register_data_check_username(), REGISTER_PASSWORD_MAX_LENGTH, magma_t::secure, st_char_get(), st_cmp_ci_eq(), st_dupe(), st_length_int(), register_session_t::username, and http_data_t::value.

Referenced by register_process().

5.459.2.7 `chr_t* register_business_step2 (connection_t * con, register_session_t * reg)`

Perform verification checking on all step 2 completed user fields, and display a welcome banner on success.

Note:

Checks include plan type validation, and billing information processing. After step 2, the user's supplied information will be persisted into the database.

Parameters:

con the underlying client connection.

reg the underlying registration session.

Returns:

NULL on success, or a descriptive error string on failure.

Definition at line 176 of file business.c.

References magma_t::admin, magma_t::contact, data, magma_t::domain, http_data_get(), HTTP_DATA_POST, log_pedantic, magma, register_session_t::password, PLACER, register_session_t::plan, register_abuse_increment_history(), register_data_insert_user(), smtp_send_message(), st_cleanup, st_cmp_cs_eq(), st_dupe(), st_merge, magma_t::system, tran_commit(), tran_rollback(), tran_start(), register_session_t::username, register_session_t::usernum, and http_data_t::value.

Referenced by register_process().

5.459.2.8 `bool_t register_business_validate_password (stringer_t * password)`

Determine whether a registered password is valid.

Note:

Each password must be between REGISTER_PASSWORD_MIN_LENGTH (5) and REGISTER_PASSWORD_MAX_LENGTH (200) characters long.

Parameters:

password the user's password to be evaluated.

Returns:

false on failure (too long, too short, or bad characters) or true on success.

Definition at line 16 of file business.c.

References length, magma, magma_t::minimum_password_length, REGISTER_PASSWORD_MAX_LENGTH, magma_t::secure, st_char_get(), st_length_get(), and utf8_length_st().

Referenced by register_business_step1().

5.459.2.9 `int_t register_business_validate_username (stringer_t * username)`

Determine whether a registered username is valid.

Parameters:

username a managed string containing the proposed username to be evaluated.

Returns:

-1 or 0 on failure (too long, too short, or bad characters) or 1 on success.

Definition at line 49 of file business.c.

References length, REGISTER_USERNAME_MAX_LENGTH, st_char_get(), and st_length_get().

Referenced by register_business_step1().

5.459.2.10 stringer_t* register_captcha_generate (stringer_t * value)

[captcha.c](#) [captcha.c](#)

Parameters:

value a managed string containing the text that is to become the basis of the captcha challenge.

Returns:

NULL on failure, or a managed string containing the path to the image file containing the captcha graphic on success.

Definition at line 106 of file captcha.c.

References gdFree_d, gdImageColorResolve_d, gdImageCreate_d, gdImageDestroy_d, gdImageGifPtr_d, gdImageStringFT_d, log_pedantic, mm_wipe(), rand_get_uint32(), register_captcha_random_font(), register_captcha_write_noise(), st_char_get(), st_import(), and st_length_get().

Referenced by register_print_captcha().

5.459.2.11 stringer_t* register_captcha_random_font (void)

LOW: We shouldn't have to actually scan the fonts directory to find a valid file. Instead we could cache a list of valid fonts and then pick from it randomly. Select a random truetype font from the directory specified in [magma.http.fonts](#).

Returns:

NULL on failure, or a managed string containing the pathname of the randomly selected font file on success.

Definition at line 15 of file captcha.c.

References count, magma_t::http, log_pedantic, magma, NULLER, PLACER, rand_get_uint32(), st_aprint(), and st_cmp_ci_ends().

Referenced by register_captcha_generate().

5.459.2.12 void register_captcha_write_noise (gdImagePtr image, int_t x, int_t y)

Fill an image's background partially with pixelated noise to make it more difficult to read.

Parameters:

image a pointer to the gd image to be modified.

x the height, in pixels, of the image region to be filled.

y the width, in pixels, of the image region to be filled.

Returns:

This function returns no value.

Definition at line 77 of file captcha.c.

References gdImageColorResolve_d, gdImageSetPixel_d, and rand_get_uint32().

Referenced by register_captcha_generate().

5.459.2.13 bool_t register_data_check_username (stringer_t * username)

datatier.c datatier.c

Parameters:

username the username to be checked against the database.

Returns:

true if the username is taken or false if it is not.

Definition at line 67 of file datatier.c.

References mm_wipe(), res_row_next(), res_table_free(), st_char_get(), st_length_get(), and stmt_get_result().

Referenced by register_business_step1().

5.459.2.14 inx_t* register_data_fetch_blocklist (void)

Fetch the blocklist for new user registration from the database. HIGH: The prepared statements being used aren't valid. The queries need to be copied over and created. register_fetch_blocklist still needs to be defined.

Returns:

an inx holder containing the registration blocklist as a collection of managed strings.

Definition at line 18 of file datatier.c.

References inx_alloc(), inx_free(), inx_insert(), inx_t, log_pedantic, M_INX_LINKED, M_TYPE_UINT64, res_field_string(), res_row_next(), res_table_free(), st_free(), stmt_get_result(), multi_t::u64, and multi_t::val.

Referenced by register_blocklist_update().

5.459.2.15 int_t register_data_insert_user (connection_t * con, uint16_t plan, stringer_t * username, stringer_t * password, int64_t transaction, uint64_t * outuser)

Insert a newly registered user into the database using information gathered by registration step #2.

Note:

The following steps occur: 1. Insert a new user into the Users table, supplying username, hashed password, plan info, and quota. 2. Insert a random shard value into the User_Realms table. 3. Insert a blank entry into the Profile table for the user. 4. Insert an entry into the Folders table for the user's "Inbox" folder. 5. Insert a new entry into the Log table containing the usernum and IP address of the client request. 6. Insert a new entry into the Dispatch table for the user, configuring the spam folder, inbox, send/receive/daily send/daily receive limits, etc. 7. Insert a new entry into the Mailboxes table for the user.

Parameters:

con a pointer to the connection object of the client making the registration request.

reg the current registration session of the user to be added.

transaction a mysql transaction id for all database operations, since they all need to be committed atomically or rolled back.

outuser a pointer to a numerical id to receive the newly generated and inserted user id.

Returns:

0 if the new user account was successfully created, -1 if a technical error occurs, and 1 if an invalid value is provided.

LOW: This function should be passed a number of days (or years) to use for any of the pre-paid account plans. LOW: The IP address could be passed in as an [ip_t](#) or as a string to avoid the need for the entire connection object.

Definition at line 114 of file `datatier.c`.

References `auth_stacie()`, `auth_stacie_cleanup()`, `base64_encode_mod()`, `con_addr_presentation()`, `magma_t::domain`, `auth_stacie_t::keys`, `log_pedantic`, `magma`, `MANAGEDBUF`, `auth_stacie_t::master`, `meta_crypto_keys_create()`, `magma_t::minimum_password_length`, `mm_wipe()`, `ns_length_get()`, `PLACER`, `magma_t::secure`, `st_char_get()`, `st_cleanup`, `st_data_get()`, `st_length_get()`, `st_length_int()`, `st_merge`, `st_search_chr()`, `st_sprint()`, `stacie_create_salt()`, `stacie_create_shard()`, `stacie_realm_key()`, `stmt_exec_conn()`, `stmt_insert_conn()`, `magma_t::system`, `time_print_local()`, `auth_stacie_t::tokens`, `utf8_length_st()`, and `auth_stacie_t::verification`.

Referenced by `api_endpoint_register()`, and `register_business_step2()`.

5.459.2.16 void register_print_captcha (connection_t * con, register_session_t * reg)

[register.c](#) [register.c](#)

Parameters:

- con** the underlying client connection.
- reg** the pending registration session of the remote client.

Returns:

This function returns no value.

Definition at line 46 of file `register.c`.

References `con_print()`, `con_write_st()`, `http_print_500()`, `register_session_t::hvf_value`, `log_info`, `rand_choices()`, `register_captcha_generate()`, `st_char_get()`, `st_free()`, and `st_length_get()`.

Referenced by `register_process()`.

5.459.2.17 void register_print_message (connection_t * con, chr_t * message)

Display a custom message to the remote client using the `register/message` template.

Parameters:

- con** the client connection to receive the message.
- message** a pointer to a null-terminated string containing the custom message to be displayed inside the template sent to the remote client.

Returns:

This function returns no value.

Definition at line 17 of file `register.c`.

References `con_write_st()`, `content`, `http_get_template()`, `http_print_500()`, `http_response_header()`, `NULLER`, `PLACER`, `http_content_t::resource`, `st_dupe()`, `st_free()`, `st_length_get()`, `st_replace()`, and `http_content_t::type`.

Referenced by `register_abuse_checks()`.

5.459.2.18 void register_print_step1 (connection_t * con, register_session_t * reg, chr_t * message)

Display step 1 of the registration process (collect username, password, and captcha challenge).

Parameters:

- con** a pointer to the connection object of the remote client making the registration request.

reg a pointer to the pending registration session of the remote client.

message if not NULL, an optional error message to be displayed inside the registration step 1 template.

Returns:

This function returns no value.

Definition at line 78 of file register.c.

References `con_write_st()`, `content`, `http_get_template()`, `http_print_500_log()`, `http_response_header()`, `register_session_t::name`, `ns_length_get()`, `PLACER`, `http_content_t::resource`, `st_dupe()`, `st_free()`, `st_length_get()`, `st_replace()`, `http_content_t::type`, and `register_session_t::username`.

Referenced by `register_process()`.

5.459.2.19 void register_print_step2 (connection_t * con, register_session_t * reg, chr_t * message)

Display step 2 of the registration process.

Note:

Previously, this step used to obtain information like the desired user plan and billing info. At the moment, however, this step effectively does nothing but display a page to be clicked-through.

Parameters:

con a pointer to the connection object of the remote client making the registration request.

reg a pointer to the pending registration session of the remote client.

message if not NULL, an optional error message to be displayed inside the registration step 2 template.

Returns:

This function returns no value.

Definition at line 125 of file register.c.

References `con_write_st()`, `content`, `http_get_template()`, `http_print_500_log()`, `http_response_header()`, `register_session_t::name`, `ns_length_get()`, `PLACER`, `http_content_t::resource`, `st_dupe()`, `st_free()`, `st_length_get()`, `st_replace()`, and `http_content_t::type`.

Referenced by `register_process()`.

5.459.2.20 void register_print_step3 (connection_t * con)

Display the registration step 3 template for a connection that has successfully created a new user.

Parameters:

the remote connection making the registration request.

Returns:

This function returns no value.

Definition at line 164 of file register.c.

References `con_write_st()`, `content`, `http_get_template()`, `http_print_500()`, `http_response_header()`, `http_content_t::resource`, `st_dupe()`, `st_free()`, `st_length_get()`, and `http_content_t::type`.

Referenced by `register_process()`.

5.459.2.21 void register_process (connection_t * con)

Process a user registration request (/register).

Note:

All requests will be redirected to a secure location. All new registration requests will also need to pass a test to ensure that they haven't been the product of abusive behavior. If an existing registration session can't be located, a new one will be generated. Depending on the user supplied http POST data, one of the three registration steps will be shown, or a captcha challenge will be presented to the user.

Returns:

This function returns no value.

Definition at line 190 of file register.c.

References con_secure(), HTTP_COMPLETE, HTTP_DATA_GET, http_data_get(), HTTP_DATA_POST, HTTP_ERROR_500, http_print_301(), log_pedantic, register_abuse_checks(), register_business_step1(), register_business_step2(), register_print_captcha(), register_print_step1(), register_print_step2(), register_print_step3(), register_session_cache(), register_session_free(), register_session_generate(), register_session_get(), register_session_t::usernum, and http_data_t::value.

Referenced by http_response().

5.459.2.22 bool_t register_session_cache (connection_t * con, register_session_t * session)

sessions.c sessions.c

Note:

The session is stored under the parent key "lavad.register.session."

Parameters:

con a pointer to the client connection underlying the user session.

session a pointer to the registration session to be persisted.

Returns:

false on failure or true if the session was cached successfully.

Definition at line 105 of file sessions.c.

References cache_set(), con_addr_presentation(), data, register_session_t::hvf_input, register_session_t::hvf_value, log_pedantic, MAN-AGEDBUF, register_session_t::name, NULLER, register_session_t::password, register_session_t::plan, serialize_st(), serialize_uint16(), serialize_uint64(), st_char_get(), st_cleanup, st_free(), st_length_int(), register_session_t::username, and register_session_t::usernum.

Referenced by register_process().

5.459.2.23 void register_session_free (register_session_t * session)

Destroy a registration session and all its associated data.

Returns:

This function returns no value.

Definition at line 14 of file sessions.c.

References register_session_t::hvf_input, register_session_t::hvf_value, mm_free(), register_session_t::name, register_session_t::password, st_cleanup, and register_session_t::username.

Referenced by register_process(), register_session_generate(), and register_session_get().

5.459.2.24 register_session_t* register_session_generate (void)

Generate a new registration session.

Returns:

NULL on failure, or a pointer to a new randomly named session on success.

Definition at line 35 of file sessions.c.

References log_pedantic, mm_alloc(), register_session_t::name, rand_choices(), and register_session_free().

Referenced by register_process().

5.459.2.25 register_session_t* register_session_get (connection_t * con, stringer_t * name)

Retrieve a registration session by a data key supplied in the user's http POST request.

Note:

The session is stored under the parent key "lavad.register.session."

Parameters:

con the client connection underlying the user request.

name a managed string containing the registration session identifier.

Returns:

NULL on failure, or a pointer to the user's registration session on success.

Definition at line 59 of file sessions.c.

References cache_get(), con_addr_presentation(), data, deserialize_st(), deserialize_uint16(), deserialize_uint64(), register_session_t::hvf_input, register_session_t::hvf_value, log_pedantic, MANAGEDDBUF, mm_alloc(), mm_wipe(), register_session_t::name, NULLER, register_session_t::password, register_session_t::plan, register_session_free(), serialization_t, st_char_get(), st_dupe(), st_free(), st_length_int(), register_session_t::username, and register_session_t::usernum.

Referenced by register_process().

5.460 src/web/statistics/statistics.h File Reference

Data Structures

- struct [statistics_vp_t](#)

Enumerations

- enum {
[portal_stat_total_users](#) = 0, [portal_stat_users_checked_email_today](#) = 1, [portal_stat_users_checked_email_week](#) = 2, [portal_stat_users_sent_email_today](#) = 3,
[portal_stat_users_sent_email_week](#) = 4, [portal_stat_emails_received_today](#) = 5, [portal_stat_emails_received_week](#) = 6, [portal_stat_emails_sent_today](#) = 7,
[portal_stat_emails_sent_week](#) = 8, [portal_stat_users_registered_today](#) = 9, [portal_stat_users_registered_week](#) = 10, [portal_stat_users_registered_total](#) = 11 }

Functions

- void [statistics_process](#) ([connection_t](#) *con)
statistics.c
- void [statistics_init](#) (void)
datatier.c
- [bool_t](#) [statistics_refresh](#) (void)
Refresh all portal statistics from the database, if they haven't been updated recently.

5.460.1 Enumeration Type Documentation

5.460.1.1 anonymous enum

Enumerator:

portal_stat_total_users
portal_stat_users_checked_email_today
portal_stat_users_checked_email_week
portal_stat_users_sent_email_today
portal_stat_users_sent_email_week
portal_stat_emails_received_today
portal_stat_emails_received_week
portal_stat_emails_sent_today
portal_stat_emails_sent_week
portal_stat_users_registered_today
portal_stat_users_registered_week
portal_stat_users_registered_total

Definition at line 11 of file statistics.h.

5.460.2 Function Documentation

5.460.2.1 void statistics_init (void)

datatier.c datatier.c

Returns:

This function returns no value.

Definition at line 22 of file datatier.c.

References mm_wipe(), portal_stat_emails_received_today, portal_stat_emails_received_week, portal_stat_emails_sent_today, portal_stat_emails_sent_week, portal_stat_total_users, portal_stat_users_checked_email_today, portal_stat_users_checked_email_week, portal_stat_users_registered_today, portal_stat_users_registered_total, portal_stat_users_registered_week, portal_stat_users_sent_email_today, portal_stat_users_sent_email_week, and statistics_vp_t::stmt.

Referenced by statistics_refresh().

5.460.2.2 void statistics_process (connection_t * con)

statistics.c statistics.c

Parameters:

con a pointer to the connection object across which the server statistics will be transmitted.

Returns:

This function returns no value.

Definition at line 17 of file statistics.c.

References con_write_st(), http_page_free(), http_page_get(), http_print_500(), http_response_header(), log_pedantic, portal_stat_emails_received_today, portal_stat_emails_received_week, portal_stat_emails_sent_today, portal_stat_emails_sent_week, portal_stat_total_users, portal_stat_users_checked_email_today, portal_stat_users_checked_email_week, portal_stat_users_registered_today, portal_stat_users_registered_week, portal_stat_users_sent_email_today, portal_stat_users_sent_email_week, st_free(), st_length_get(), statistics_refresh(), statistics_vp_t::val, xml_dump_doc(), xml_set_xpath_ns(), and xml_set_xpath_uint64().

Referenced by http_response().

5.460.2.3 bool_t statistics_refresh (void)

Refresh all portal statistics from the database, if they haven't been updated recently.

Returns:

true if all statistics were refreshed successfully, or false if they were not.

Definition at line 44 of file datatier.c.

References log_pedantic, mutex_lock(), mutex_unlock(), portal_statistics_mutex, PORTAL_STATISTICS_TIMEOUT, res_field_uint64(), res_row_next(), res_table_free(), statistics_init(), statistics_last_updated, statistics_vp_t::stmt, stmt_get_result(), and statistics_vp_t::val.

Referenced by statistics_process().

5.461 src/web/teacher/teacher.c File Reference

```
#include "magma.h"
```

Functions

- void [teacher_print_message](#) ([connection_t](#) *con, [chr_t](#) *message)
Display a custom message to the remote client using the teacher/message template.
- void [teacher_print_form](#) ([connection_t](#) *con, [http_page_t](#) *page, [teacher_data_t](#) *teach)
Display a custom message to the remote client using the teacher/message template.
- [http_page_t](#) * [teacher_add_error](#) ([chr_t](#) *xpath, [chr_t](#) *id, [chr_t](#) *message)
Get the teacher/teacher template and add an error message to it.
- void [teacher_add_cookie](#) ([connection_t](#) *con, [teacher_data_t](#) *teach)
Create a cookie for a successfully password-authenticated teacher request.
- void [teacher_process](#) ([connection_t](#) *con)
The main entry point for the /teacher web application.

5.461.1 Function Documentation

5.461.1.1 void [teacher_add_cookie](#) ([connection_t](#) * con, [teacher_data_t](#) * teach)

Create a cookie for a successfully password-authenticated teacher request. [teacher.c](#)

Parameters:

- con** the connection of the web client accessing the teacher facility.
- teach** the spam signature associated with the cookie request; it supplies the identifying password stored in the cookie.

Returns:

This function returns no value.

Definition at line 146 of file [teacher.c](#).

References [inx_alloc\(\)](#), [inx_insert\(\)](#), [log_pedantic](#), [M_INX_LINKED](#), [M_TYPE_STRINGER](#), [ns_length_get\(\)](#), [teacher_data_t::password](#), [multi_t::st](#), [st_char_get\(\)](#), [st_free\(\)](#), [st_import\(\)](#), and [multi_t::val](#).

Referenced by [teacher_process\(\)](#).

5.461.1.2 [http_page_t](#)* [teacher_add_error](#) ([chr_t](#) * xpath, [chr_t](#) * id, [chr_t](#) * message)

Get the teacher/teacher template and add an error message to it.

Parameters:

- xpath** a null-terminated string containing the xpath of the node in the template to be marked with the error message.
- id** a null-terminated string containing the id of the error message node to be added as the sibling of the node in the specified xpath.
- message** a null-terminated string containing the error message to be displayed to the user.
- NULL** on failure, or a pointer to the modified teacher/teacher template page on success.

Definition at line 107 of file teacher.c.

References `http_page_get()`, `xml_node_add_sibling()`, `xml_node_free()`, `xml_node_new()`, `xml_node_set_content()`, `xml_node_set_property()`, `xml_xpath_eval()`, and `http_page_t::xpath_ctx`.

Referenced by `teacher_process()`.

5.461.1.3 void teacher_print_form (connection_t * con, http_page_t * page, teacher_data_t * teach)

Display a custom message to the remote client using the teacher/message template.

Parameters:

con the client connection to receive the message.

message a null-terminated string containing the custom message to be displayed inside the template sent to the remote client.

Returns:

This function returns no value.

Definition at line 63 of file teacher.c.

References `con_write_st()`, `http_page_t::content`, `teacher_data_t::disposition`, `http_page_t::doc_obj`, `http_page_free()`, `http_page_get()`, `http_print_500()`, `http_response_header()`, `teacher_data_t::keynum`, `teacher_data_t::signum`, `st_free()`, `st_length_get()`, `teacher_print_message()`, `http_content_t::type`, `xml_dump_doc()`, `xml_set_xpath_ns()`, `xml_set_xpath_property()`, and `http_page_t::xpath_ctx`.

Referenced by `teacher_process()`.

5.461.1.4 void teacher_print_message (connection_t * con, chr_t * message)

Display a custom message to the remote client using the teacher/message template.

Parameters:

con the client connection to receive the message.

message a null-terminated string containing the custom message to be displayed inside the template sent to the remote client.

Returns:

This function returns no value.

Definition at line 16 of file teacher.c.

References `con_print()`, `con_write_st()`, `http_page_t::content`, `http_page_t::doc_obj`, `http_page_free()`, `http_page_get()`, `http_print_500()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `st_char_get()`, `st_free()`, `st_length_get()`, `http_content_t::type`, `xml_dump_doc()`, `xml_set_xpath_ns()`, and `http_page_t::xpath_ctx`.

Referenced by `teacher_print_form()`, and `teacher_process()`.

5.461.1.5 void teacher_process (connection_t * con)

The main entry point for the /teacher web application.

Note:

Each /teacher request must contain a spam signature and key specified by the user. If Magma was able to find the signature, the signature's key must match the user-specified key value. The signature must also not have been previously trained. On successful training, the signature will be freed in memory and in the database, but kept for a couple of hours in distributed cache. User password authentication is necessary for training, and this routine manages cookies for subsequent training requests.

Parameters:

con the connection object underlying the client /teacher request.

Returns:

This function returns no value.

TODO FIXME: Memory leaks here. This part isn't fixed to work with the new credentials functions, so be careful reusing this code.

Definition at line 220 of file teacher.c.

References `auth_challenge()`, `auth_free()`, `teacher_data_t::completed`, `con_secure()`, `teacher_data_t::disposition`, `dspam_train()`, `HTTP_COMPLETE`, `HTTP_DATA_ANY`, `http_data_get()`, `HTTP_DATA_HEADER`, `HTTP_DATA_POST`, `http_print_301()`, `teacher_data_t::keynum`, `teacher_data_t::password`, `PLACER`, `teacher_data_t::signature`, `st_char_get()`, `st_cleanup`, `st_cmp_cs_eq()`, `st_cmp_cs_starts()`, `st_dupe()`, `st_length_get()`, `st_replace()`, `teacher_add_cookie()`, `teacher_add_error()`, `teacher_data_delete()`, `teacher_data_free()`, `teacher_data_get()`, `teacher_data_save()`, `teacher_print_form()`, `teacher_print_message()`, `auth_t::tokens`, `uint64_conv_st()`, `teacher_data_t::username`, `teacher_data_t::usernum`, `http_data_t::value`, and `auth_t::verification`.

Referenced by `http_response()`.

5.462 src/web/teacher/teacher.h File Reference

Data Structures

- struct [teacher_data_t](#)

Functions

- void [teacher_data_delete](#) ([teacher_data_t](#) *teach)
datatier.c
- [teacher_data_t](#) * [teacher_data_fetch](#) (uint64_t signum)
Fetch information about a spam signature from the database.
- void [teacher_data_free](#) ([teacher_data_t](#) *teach)
Free a spam signature object.
- [teacher_data_t](#) * [teacher_data_get](#) (uint64_t signum)
Get information about a spam signature from the cache, or fall back to the database.
- void [teacher_data_save](#) ([teacher_data_t](#) *teach)
Save spam signature information to the cache.
- void [teacher_add_cookie](#) ([connection_t](#) *con, [teacher_data_t](#) *teach)
teacher.c
- [http_page_t](#) * [teacher_add_error](#) ([chr_t](#) *xpath, [chr_t](#) *id, [chr_t](#) *message)
Get the teacher/teacher template and add an error message to it.
- void [teacher_print_form](#) ([connection_t](#) *con, [http_page_t](#) *page, [teacher_data_t](#) *teach)
Display a custom message to the remote client using the teacher/message template.
- void [teacher_print_message](#) ([connection_t](#) *con, [chr_t](#) *message)
Display a custom message to the remote client using the teacher/message template.
- void [teacher_process](#) ([connection_t](#) *con)
The main entry point for the /teacher web application.

5.462.1 Function Documentation

5.462.1.1 void [teacher_add_cookie](#) ([connection_t](#) * con, [teacher_data_t](#) * teach)

[teacher.c](#) [teacher.c](#)

Parameters:

- con** the connection of the web client accessing the teacher facility.
- teach** the spam signature associated with the cookie request; it supplies the identifying password stored in the cookie.

Returns:

This function returns no value.

Definition at line 146 of file teacher.c.

References `inx_alloc()`, `inx_insert()`, `log_pedantic`, `M_INX_LINKED`, `M_TYPE_STRINGER`, `ns_length_get()`, `teacher_data_t::password`, `multi_t::st`, `st_char_get()`, `st_free()`, `st_import()`, and `multi_t::val`.

Referenced by `teacher_process()`.

5.462.1.2 `http_page_t* teacher_add_error (chr_t * xpath, chr_t * id, chr_t * message)`

Get the teacher/teacher template and add an error message to it.

Parameters:

xpath a null-terminated string containing the xpath of the node in the template to be marked with the error message.

id a null-terminated string containing the id of the error message node to be added as the sibling of the node in the specified xpath.

message a null-terminated string containing the error message to be displayed to the user.

NULL on failure, or a pointer to the modified teacher/teacher template page on success.

Definition at line 107 of file teacher.c.

References `http_page_get()`, `xml_node_add_sibling()`, `xml_node_free()`, `xml_node_new()`, `xml_node_set_content()`, `xml_node_set_property()`, `xml_xpath_eval()`, and `http_page_t::xpath_ctx`.

Referenced by `teacher_process()`.

5.462.1.3 `void teacher_data_delete (teacher_data_t * teach)`

datatier.c datatier.c

HIGH: After training a signature, we should search the messages table for references to the signature being trained and update the message status flags. The `UPDATE_SIGNATURE_FLAGS_ADD/UPDATE_SIGNATURE_FLAGS_REMOVE` queries were created for that purpose but aren't being used right now. Note to self: retroactively brand messages as junk accordingly.

Note:

If the signature matched junk, all matching messages in the database belonging to the user will have their junk flag cleared. But if the signature didn't match junk, all matching messages in the database belonging to the user will have the junk flag added.

Parameters:

teach the spam signature to be removed from the database.

Returns:

This function returns no value.

Definition at line 38 of file datatier.c.

References `teacher_data_t::completed`, `teacher_data_t::disposition`, `log_pedantic`, `MAIL_MARK_JUNK`, `mm_wipe()`, `teacher_data_t::password`, `teacher_data_t::signature`, `teacher_data_t::signum`, `st_cleanup`, `stmt_exec()`, `teacher_data_t::username`, and `teacher_data_t::usernum`.

Referenced by `teacher_process()`.

5.462.1.4 `teacher_data_t* teacher_data_fetch (uint64_t signum)`

Fetch information about a spam signature from the database.

Parameters:

signum the numerical id of the spam signature to be retrieved.

NULL on failure, or a pointer to a newly allocated signature teacher object on success.

Definition at line 137 of file datatier.c.

References `teacher_data_t::disposition`, `teacher_data_t::keynum`, `mm_alloc()`, `mm_wipe()`, `teacher_data_t::password`, `res_field_int8()`, `res_field_string()`, `res_field_uint64()`, `res_row_next()`, `res_table_free()`, `teacher_data_t::signature`, `teacher_data_t::signum`, `stmt_get_result()`, `teacher_data_free()`, `teacher_data_t::username`, and `teacher_data_t::usernum`.

Referenced by `teacher_data_get()`.

5.462.1.5 void teacher_data_free (teacher_data_t * teach)

Free a spam signature object.

Parameters:

teach the spam signature object to be freed.

Returns:

This function returns no value.

Definition at line 15 of file datatier.c.

References `mm_free()`, `teacher_data_t::password`, `teacher_data_t::signature`, `st_cleanup`, and `teacher_data_t::username`.

Referenced by `teacher_data_fetch()`, `teacher_data_get()`, and `teacher_process()`.

5.462.1.6 teacher_data_t* teacher_data_get (uint64_t signum)

Get information about a spam signature from the cache, or fall back to the database.

Parameters:

signum the numerical id of the spam signature to be retrieved.

NULL on failure, or a pointer to a newly allocated signature teacher object on success.

Definition at line 229 of file datatier.c.

References `cache_get()`, `teacher_data_t::completed`, `data`, `deserialize_int32()`, `deserialize_st()`, `deserialize_uint64()`, `teacher_data_t::disposition`, `teacher_data_t::keynum`, `log_pedantic`, `mm_alloc()`, `mm_wipe()`, `teacher_data_t::password`, `PLACER`, `serialization_t`, `teacher_data_t::signature`, `teacher_data_t::signum`, `st_free()`, `teacher_data_fetch()`, `teacher_data_free()`, `teacher_data_t::username`, and `teacher_data_t::usernum`.

Referenced by `teacher_process()`.

5.462.1.7 void teacher_data_save (teacher_data_t * teach)

Save spam signature information to the cache.

Note:

The information will be cached for 2 hours.

Parameters:

teach the spam signature to be cached.

Returns:

This function returns no value.

Definition at line 189 of file `datatier.c`.

References `cache_set()`, `teacher_data_t::completed`, `data`, `teacher_data_t::disposition`, `teacher_data_t::keynum`, `log_pedantic`, `teacher_data_t::password`, `PLACER`, `serialize_int32()`, `serialize_st()`, `serialize_uint64()`, `teacher_data_t::signature`, `teacher_data_t::signum`, `st_free()`, `teacher_data_t::username`, and `teacher_data_t::usernum`.

Referenced by `teacher_process()`.

5.462.1.8 `void teacher_print_form (connection_t * con, http_page_t * page, teacher_data_t * teach)`

Display a custom message to the remote client using the `teacher/message` template.

Parameters:

con the client connection to receive the message.

message a null-terminated string containing the custom message to be displayed inside the template sent to the remote client.

Returns:

This function returns no value.

Definition at line 63 of file `teacher.c`.

References `con_write_st()`, `http_page_t::content`, `teacher_data_t::disposition`, `http_page_t::doc_obj`, `http_page_free()`, `http_page_get()`, `http_print_500()`, `http_response_header()`, `teacher_data_t::keynum`, `teacher_data_t::signum`, `st_free()`, `st_length_get()`, `teacher_print_message()`, `http_content_t::type`, `xml_dump_doc()`, `xml_set_xpath_ns()`, `xml_set_xpath_property()`, and `http_page_t::xpath_ctx`.

Referenced by `teacher_process()`.

5.462.1.9 `void teacher_print_message (connection_t * con, chr_t * message)`

Display a custom message to the remote client using the `teacher/message` template.

Parameters:

con the client connection to receive the message.

message a null-terminated string containing the custom message to be displayed inside the template sent to the remote client.

Returns:

This function returns no value.

Definition at line 16 of file `teacher.c`.

References `con_print()`, `con_write_st()`, `http_page_t::content`, `http_page_t::doc_obj`, `http_page_free()`, `http_page_get()`, `http_print_500()`, `inx_cursor_alloc()`, `inx_cursor_free()`, `inx_cursor_t`, `inx_cursor_value_next()`, `st_char_get()`, `st_free()`, `st_length_get()`, `http_content_t::type`, `xml_dump_doc()`, `xml_set_xpath_ns()`, and `http_page_t::xpath_ctx`.

Referenced by `teacher_print_form()`, and `teacher_process()`.

5.462.1.10 `void teacher_process (connection_t * con)`

The main entry point for the `/teacher` web application.

Note:

Each `/teacher` request must contain a spam signature and key specified by the user. If Magma was able to find the signature, the signature's key must match the user-specified key value. The signature must also not have been previously trained. On successful training, the signature will be freed in memory and in the database, but kept for a couple of hours in distributed cache. User password authentication is necessary for training, and this routine manages cookies for subsequent training requests.

Parameters:

con the connection object underlying the client /teacher request.

Returns:

This function returns no value.

TODO FIXME: Memory leaks here. This part isn't fixed to work with the new credentials functions, so be careful reusing this code.

Definition at line 220 of file teacher.c.

References `auth_challenge()`, `auth_free()`, `teacher_data_t::completed`, `con_secure()`, `teacher_data_t::disposition`, `dspam_train()`, `HTTP_COMPLETE`, `HTTP_DATA_ANY`, `http_data_get()`, `HTTP_DATA_HEADER`, `HTTP_DATA_POST`, `http_print_301()`, `teacher_data_t::keynum`, `teacher_data_t::password`, `PLACER`, `teacher_data_t::signature`, `st_char_get()`, `st_cleanup`, `st_cmp_cs_eq()`, `st_cmp_cs_starts()`, `st_dupe()`, `st_length_get()`, `st_replace()`, `teacher_add_cookie()`, `teacher_add_error()`, `teacher_data_delete()`, `teacher_data_free()`, `teacher_data_get()`, `teacher_data_save()`, `teacher_print_form()`, `teacher_print_message()`, `auth_t::tokens`, `uint64_conv_st()`, `teacher_data_t::username`, `teacher_data_t::usernum`, `http_data_t::value`, and `auth_t::verification`.

Referenced by `http_response()`.

5.463 src/web/web.h File Reference

```
#include "contact/contact.h"
#include "json_api/json_api.h"
#include "portal/portal.h"
#include "register/register.h"
#include "statistics/statistics.h"
#include "teacher/teacher.h"
```

Index

- [__VA_NARG_N](#)
 - [strings.h, 539](#)
- [__VA_NARG_SEQ_N](#)
 - [strings.h, 539](#)
- [__VA_NARG__](#)
 - [strings.h, 539](#)
- [__attribute__](#), 17
 - [aes.c, 1856](#)
 - [agent, 18](#)
 - [binary.h, 1925](#)
 - [body, 18](#)
 - [buckets.h, 169](#)
 - [compress.h, 1294](#)
 - [connection, 18](#)
 - [consumers.h, 1309](#)
 - [contacts.h, 1045](#)
 - [cookie, 18](#)
 - [core.h, 230](#)
 - [crypto.h, 1583](#)
 - [cryptography/cryptography.h, 1350](#)
 - [dmsg.c, 1592](#)
 - [dns.h, 1750](#)
 - [error.c, 1520](#)
 - [error.h, 1533](#)
 - [expunge, 18](#)
 - [folders.h, 1075](#)
 - [hashed.c, 331](#)
 - [headers, 18](#)
 - [host, 18](#)
 - [id, 18](#)
 - [indexes.h, 338](#)
 - [linked.c, 356](#)
 - [location, 18](#)
 - [merged, 18](#)
 - [method, 19](#)
 - [mode, 19](#)
 - [network/imap.h, 872](#)
 - [network/meta.h, 907](#)
 - [network/network.h, 934](#)
 - [network/sessions.h, 971](#)
 - [nvp_t, 115](#)
 - [objects/config/config.h, 678](#)
 - [objects/messages/messages.h, 1179](#)
 - [pairs, 19](#)
 - [params, 19](#)
 - [port, 19](#)
 - [portal, 19](#)
 - [prime.h, 1906](#)
 - [providers.h, 1938](#)
 - [request, 19](#)
 - [response, 19](#)
 - [sds.h, 1701](#)
 - [secure.c, 387](#)
 - [session, 19](#)
 - [session_state, 19](#)
 - [storage.h, 1959](#)
 - [strings.h, 543](#)
 - [tree.c, 1972](#)
 - [user, 19](#)
 - [username, 19](#)
 - [usernum, 20](#)
 - [warehouse.h, 1253](#)
- [__bool_t_defined](#)
 - [core.h, 225](#)
- [__dbgprint](#)
 - [misc.c, 1538](#)
- [__failed_tests](#)
 - [testhelp.h, 1707](#)
- [__int24_t](#)
 - [core.h, 231](#)
- [__lzo_init_v2_d](#)
 - [symbols.c, 1984](#)
 - [symbols.h, 2023](#)
- [__str_printf](#)
 - [misc.c, 1538](#)
- [__test_num](#)
 - [testhelp.h, 1707](#)
- [__uint24_t](#)
 - [core.h, 231](#)
- [_add_cached_object](#)
 - [providers/dime/signet-resolver/cache.c, 627](#)
- [_add_cached_object_cmp](#)
 - [providers/dime/signet-resolver/cache.c, 627](#)
- [_add_cached_object_cmp_forced](#)
 - [providers/dime/signet-resolver/cache.c, 628](#)
- [_add_cached_object_forced](#)
 - [providers/dime/signet-resolver/cache.c, 628](#)
- [_add_dnskey_entry](#)
 - [dns.c, 1732](#)
 - [dns.h, 1751](#)
- [_add_dnskey_entry_rsa](#)
 - [dns.c, 1733](#)
 - [dns.h, 1751](#)
- [_add_ds_entry](#)
 - [dns.c, 1733](#)
 - [dns.h, 1751](#)

- `_b64decode`
 - `misc.c`, 1538
- `_b64decode_nopad`
 - `misc.c`, 1539
- `_b64encode`
 - `misc.c`, 1539
- `_b64encode_nopad`
 - `misc.c`, 1539
- `_b64encode_w_lineseparators`
 - `misc.c`, 1540
- `_cached_object_exists`
 - `providers/dime/signet-resolver/cache.c`, 629
- `_cached_object_exists_cmp`
 - `providers/dime/signet-resolver/cache.c`, 629
- `_clear_error_stack`
 - `error.c`, 1520
 - `error.h`, 1533
- `_clone_cached_object`
 - `providers/dime/signet-resolver/cache.c`, 629
 - `providers/dime/signet-resolver/cache.h`, 652
- `_clone_dnskey_record_cb`
 - `dns.c`, 1734
 - `dns.h`, 1752
- `_compare_rdata`
 - `dns.c`, 1734
 - `dns.h`, 1752
- `_compute_aes256_kek`
 - `providers/dime/common/crypto.c`, 1199
- `_compute_crc24_checksum`
 - `misc.c`, 1540
- `_compute_dnskey_sha_hash`
 - `dns.c`, 1734
- `_compute_sha_hash`
 - `misc.c`, 1540
- `_compute_sha_hash_multibuf`
 - `misc.c`, 1541
- `_connect_host`
 - `network.c`, 1561
- `_connect_timeout`
 - `network.c`, 1561
 - `providers/dime/common/network.h`, 951
- `_count_ptr_chain`
 - `misc.c`, 1541
- `_create_cached_object`
 - `providers/dime/signet-resolver/cache.c`, 630
 - `providers/dime/signet-resolver/cache.h`, 652
- `_create_new_error`
 - `error.c`, 1520
 - `error.h`, 1533
- `_crypto_init`
 - `providers/dime/common/crypto.c`, 1199
- `_crypto_shutdown`
 - `providers/dime/common/crypto.c`, 1199
- `_dbgprint`
 - `misc.c`, 1541
- `_decode_rsa_pubkey`
 - `misc.c`, 1542
- `_decrypt_aes_256`
 - `providers/dime/common/crypto.c`, 1199
- `_deserialize_array`
 - `providers/dime/signet-resolver/cache.c`, 630
 - `providers/dime/signet-resolver/cache.h`, 653
- `_deserialize_array_cb`
 - `providers/dime/signet-resolver/cache.c`, 631
 - `providers/dime/signet-resolver/cache.h`, 653
- `_deserialize_data`
 - `providers/dime/signet-resolver/cache.c`, 631
 - `providers/dime/signet-resolver/cache.h`, 654
- `_deserialize_dime_record_cb`
 - `mrec.c`, 1760
 - `mrec.h`, 1765
- `_deserialize_dnskey_record_cb`
 - `dns.c`, 1735
 - `dns.h`, 1752
- `_deserialize_ds_record_cb`
 - `dns.c`, 1735
 - `dns.h`, 1753
- `_deserialize_ec_privkey`
 - `providers/dime/common/crypto.c`, 1199
- `_deserialize_ec_pubkey`
 - `providers/dime/common/crypto.c`, 1200
- `_deserialize_ed25519_privkey`
 - `providers/dime/common/crypto.c`, 1200
- `_deserialize_ed25519_pubkey`
 - `providers/dime/common/crypto.c`, 1200
- `_deserialize_ocsp_response_cb`
 - `signet-ssl.h`, 1770
 - `ssl.c`, 1776
- `_deserialize_signet_cb`
 - `providers/dime/signet-resolver/cache.c`, 632
 - `providers/dime/signet-resolver/cache.h`, 654
- `_deserialize_str_array`
 - `providers/dime/signet-resolver/cache.c`, 632
 - `providers/dime/signet-resolver/cache.h`, 654
- `_deserialize_string`
 - `providers/dime/signet-resolver/cache.c`, 632
 - `providers/dime/signet-resolver/cache.h`, 655
- `_deserialize_vardata`
 - `providers/dime/signet-resolver/cache.c`, 633
 - `providers/dime/signet-resolver/cache.h`, 655
- `_destroy_cache_entry`
 - `providers/dime/signet-resolver/cache.c`, 633
- `_destroy_dime_record`
 - `mrec.c`, 1761
- `_destroy_dime_record_cb`
 - `mrec.c`, 1761
 - `mrec.h`, 1766
- `_destroy_dnskey`
 - `dns.c`, 1735
- `_destroy_dnskey_record_cb`
 - `dns.c`, 1735
 - `dns.h`, 1753
- `_destroy_ds`
 - `dns.c`, 1736

- `_destroy_ds_record_cb`
 - `dns.c`, [1736](#)
 - `dns.h`, [1753](#)
- `_destroy_ocsp_response_cb`
 - `signet-ssl.h`, [1770](#)
 - `ssl.c`, [1776](#)
- `_destroy_signet_cb`
 - `providers/dime/signet-resolver/cache.c`, [634](#)
 - `providers/dime/signet-resolver/cache.h`, [656](#)
- `_dnskey_domain_comparator`
 - `dns.c`, [1736](#)
 - `dns.h`, [1754](#)
- `_dnskey_tag_comparator`
 - `dns.c`, [1736](#)
 - `dns.h`, [1754](#)
- `_do_ocsp_validation`
 - `ssl.c`, [1777](#)
- `_do_x509_hostname_check`
 - `ssl.c`, [1777](#)
- `_do_x509_validation`
 - `ssl.c`, [1778](#)
- `_domain_wildcard_check`
 - `signet-ssl.h`, [1771](#)
 - `ssl.c`, [1778](#)
- `_ds_comparator`
 - `dns.c`, [1737](#)
 - `dns.h`, [1754](#)
- `_dump_buf`
 - `misc.c`, [1542](#)
- `_dump_buf_outter`
 - `misc.c`, [1542](#)
- `_dump_cache`
 - `providers/dime/signet-resolver/cache.c`, [634](#)
 - `providers/dime/signet-resolver/cache.h`, [656](#)
- `_dump_cache_data`
 - `providers/dime/signet-resolver/cache.c`, [634](#)
 - `providers/dime/signet-resolver/cache.h`, [656](#)
- `_dump_dime_record_cb`
 - `mrec.c`, [1761](#)
 - `mrec.h`, [1766](#)
- `_dump_dns_header`
 - `dns.c`, [1737](#)
 - `dns.h`, [1754](#)
- `_dump_dnskey_record_cb`
 - `dns.c`, [1737](#)
 - `dns.h`, [1755](#)
- `_dump_ds_record_cb`
 - `dns.c`, [1738](#)
 - `dns.h`, [1755](#)
- `_dump_error`
 - `error.c`, [1520](#)
 - `error.h`, [1533](#)
- `_dump_ocsp_response_cb`
 - `signet-ssl.h`, [1771](#)
 - `ssl.c`, [1778](#)
- `_dump_signet_cb`
 - `providers/dime/signet-resolver/cache.c`, [634](#)
 - `providers/dime/signet-resolver/cache.h`, [657](#)
- `_dx_connect_dual`
 - `providers/dime/signet-resolver/dmtp.c`, [1712](#)
- `_dx_connect_standard`
 - `providers/dime/signet-resolver/dmtp.c`, [1713](#)
- `_ec_sign_data`
 - `providers/dime/common/crypto.c`, [1201](#)
- `_ec_sign_sha_data`
 - `providers/dime/common/crypto.c`, [1201](#)
- `_ecies_env_derivation`
 - `providers/dime/common/crypto.c`, [1201](#)
- `_ed25519_sign_data`
 - `providers/dime/common/crypto.c`, [1202](#)
- `_ed25519_verify_sig`
 - `providers/dime/common/crypto.c`, [1202](#)
- `_encode_rsa_pubkey`
 - `misc.c`, [1543](#)
- `_encrypt_aes_256`
 - `providers/dime/common/crypto.c`, [1202](#)
- `_evict_if_stale`
 - `providers/dime/signet-resolver/cache.c`, [635](#)
 - `providers/dime/signet-resolver/cache.h`, [657](#)
- `_fd`
 - `dmtp_session_t`, [54](#)
- `_find_cached_object`
 - `providers/dime/signet-resolver/cache.c`, [635](#)
- `_find_cached_object_cmp`
 - `providers/dime/signet-resolver/cache.c`, [635](#)
- `_fixup_dnskey_validation`
 - `dns.c`, [1738](#)
 - `dns.h`, [1755](#)
- `_fixup_ds_links`
 - `dns.c`, [1738](#)
 - `dns.h`, [1756](#)
- `_free_ec_key`
 - `providers/dime/common/crypto.c`, [1202](#)
- `_free_ed25519_key`
 - `providers/dime/common/crypto.c`, [1203](#)
- `_free_ed25519_key_chain`
 - `providers/dime/common/crypto.c`, [1203](#)
- `_free_mx_records`
 - `dns.c`, [1738](#)
- `_generate_ec_keypair`
 - `providers/dime/common/crypto.c`, [1203](#)
- `_generate_ed25519_keypair`
 - `providers/dime/common/crypto.c`, [1203](#)
- `_get_cache_location`
 - `providers/dime/signet-resolver/cache.c`, [636](#)
- `_get_cache_obj_data`
 - `providers/dime/signet-resolver/cache.c`, [636](#)
- `_get_cache_ocsp_id`
 - `signet-ssl.h`, [1771](#)
 - `ssl.c`, [1779](#)
- `_get_cached_store_by_type`
 - `providers/dime/signet-resolver/cache.c`, [636](#)
 - `providers/dime/signet-resolver/cache.h`, [657](#)
- `_get_cert_store`

- signet-ssl.h, 1772
- ssl.c, 1779
- _get_cert_subject_cn
 - ssl.c, 1779
- _get_chr_date
 - misc.c, 1543
- _get_dbg_level
 - misc.c, 1543
- _get_dime_dir_location
 - providers/dime/signet-resolver/cache.c, 637
- _get_dime_record
 - mrec.c, 1761
- _get_dime_record_from_file
 - mrec.c, 1762
- _get_dnskey_by_tag
 - dns.c, 1739
- _get_ds_by_dnskey
 - dns.c, 1739
- _get_keytag
 - dns.c, 1739
- _get_mx_records
 - dns.c, 1740
- _get_random_bytes
 - providers/dime/common/crypto.c, 1203
- _get_rsa_dnskey
 - dns.c, 1740
 - dns.h, 1756
- _get_signet
 - providers/dime/signet-resolver/dmtp.c, 1713
- _get_signing_key_names
 - dns.c, 1740
 - dns.h, 1756
- _get_txt_record
 - dns.c, 1741
- _get_x509_cert_sha_hash
 - misc.c, 1544
- _hex_encode
 - misc.c, 1544
- _inbuf
 - dmtp_session_t, 54
- _initialize_resolver
 - dns.c, 1741
 - dns.h, 1756
- _inpos
 - dmtp_session_t, 54
- _int_no_get_2b
 - misc.c, 1544
- _int_no_get_3b
 - misc.c, 1545
- _int_no_get_4b
 - misc.c, 1545
- _int_no_put_2b
 - misc.c, 1545
- _int_no_put_3b
 - misc.c, 1545
- _int_no_put_4b
 - misc.c, 1545
- _is_buf_zeroed
 - misc.c, 1546
- _is_object_expired
 - providers/dime/signet-resolver/cache.c, 637
 - providers/dime/signet-resolver/cache.h, 657
- _is_validated_key
 - dns.c, 1741
- _load_cache_contents
 - providers/dime/signet-resolver/cache.c, 637
- _load_dnskey_file
 - dns.c, 1742
- _load_ec_privkey
 - providers/dime/common/crypto.c, 1204
- _load_ec_pubkey
 - providers/dime/common/crypto.c, 1204
- _load_ed25519_privkey
 - providers/dime/common/crypto.c, 1204
- _lock_cache_store
 - providers/dime/signet-resolver/cache.c, 638
 - providers/dime/signet-resolver/cache.h, 658
- _lookup_dnskey
 - dns.c, 1742
- _lookup_ds
 - dns.c, 1742
- _mem_append
 - misc.c, 1546
- _mem_append_canon
 - dns.c, 1743
 - dns.h, 1757
- _mem_append_serialized
 - providers/dime/signet-resolver/cache.c, 638
 - providers/dime/signet-resolver/cache.h, 658
- _mem_append_serialized_array
 - providers/dime/signet-resolver/cache.c, 638
 - providers/dime/signet-resolver/cache.h, 658
- _mem_append_serialized_array_cb
 - providers/dime/signet-resolver/cache.c, 639
 - providers/dime/signet-resolver/cache.h, 659
- _mem_append_serialized_str_array
 - providers/dime/signet-resolver/cache.c, 639
 - providers/dime/signet-resolver/cache.h, 659
- _mem_append_serialized_string
 - providers/dime/signet-resolver/cache.c, 639
 - providers/dime/signet-resolver/cache.h, 660
- _ocsp_response_callback
 - signet-ssl.h, 1772
 - ssl.c, 1780
- _parse_dime_record
 - mrec.c, 1762
- _ptr_chain_add
 - misc.c, 1546
- _ptr_chain_clone
 - misc.c, 1547
- _ptr_chain_free
 - misc.c, 1547
- _push_error_stack
 - error.c, 1521

- error.h, 1533
- _push_error_stack_fmt
 - error.c, 1521
 - error.h, 1533
- _push_error_stack_openssl
 - error.c, 1521
 - error.h, 1534
- _push_error_stack_resolver
 - error.c, 1521
 - error.h, 1534
- _push_error_stack_syscall
 - error.c, 1521
 - error.h, 1534
- _read_file_data
 - misc.c, 1547
- _read_pem_data
 - misc.c, 1548
- _remove_cached_object
 - providers/dime/signet-resolver/cache.c, 640
- _remove_cached_object_cmp
 - providers/dime/signet-resolver/cache.c, 640
- _replace_object
 - providers/dime/signet-resolver/cache.c, 641
 - providers/dime/signet-resolver/cache.h, 660
- _rsa_verify_record
 - dns.c, 1743
- _save_cache_contents
 - providers/dime/signet-resolver/cache.c, 641
- _secure_wipe
 - misc.c, 1548
- _serialize_dime_record_cb
 - mrec.c, 1763
 - mrec.h, 1766
- _serialize_dnskey_record_cb
 - dns.c, 1744
 - dns.h, 1757
- _serialize_ds_record_cb
 - dns.c, 1744
 - dns.h, 1757
- _serialize_ec_privkey
 - providers/dime/common/crypto.c, 1205
- _serialize_ec_pubkey
 - providers/dime/common/crypto.c, 1205
- _serialize_ocsp_response_cb
 - signet-ssl.h, 1772
 - ssl.c, 1780
- _serialize_signet_cb
 - providers/dime/signet-resolver/cache.c, 641
 - providers/dime/signet-resolver/cache.h, 660
- _set_cache_location
 - providers/dime/signet-resolver/cache.c, 642
- _set_cache_permissions
 - providers/dime/signet-resolver/cache.c, 642
- _set_dbg_level
 - misc.c, 1548
- _sgnt_resolv_destroy_dmtplib_session
 - providers/dime/signet-resolver/dmtplib.c, 1713
- _sgnt_resolv_dmtplib_connect
 - providers/dime/signet-resolver/dmtplib.c, 1714
- _sgnt_resolv_dmtplib_data
 - providers/dime/signet-resolver/dmtplib.c, 1714
- _sgnt_resolv_dmtplib_ahlo
 - providers/dime/signet-resolver/dmtplib.c, 1714
- _sgnt_resolv_dmtplib_expect_banner
 - providers/dime/signet-resolver/dmtplib.c, 1715
 - providers/dime/signet-resolver/dmtplib.h, 838
- _sgnt_resolv_dmtplib_get_mode
 - providers/dime/signet-resolver/dmtplib.c, 1715
- _sgnt_resolv_dmtplib_get_signet
 - providers/dime/signet-resolver/dmtplib.c, 1715
- _sgnt_resolv_dmtplib_help
 - providers/dime/signet-resolver/dmtplib.c, 1715
- _sgnt_resolv_dmtplib_history
 - providers/dime/signet-resolver/dmtplib.c, 1716
- _sgnt_resolv_dmtplib_initiate_starttls
 - providers/dime/signet-resolver/dmtplib.c, 1716
 - providers/dime/signet-resolver/dmtplib.h, 839
- _sgnt_resolv_dmtplib_issue_command
 - providers/dime/signet-resolver/dmtplib.c, 1716
 - providers/dime/signet-resolver/dmtplib.h, 839
- _sgnt_resolv_dmtplib_mail_from
 - providers/dime/signet-resolver/dmtplib.c, 1717
- _sgnt_resolv_dmtplib_noop
 - providers/dime/signet-resolver/dmtplib.c, 1717
- _sgnt_resolv_dmtplib_quit
 - providers/dime/signet-resolver/dmtplib.c, 1717
- _sgnt_resolv_dmtplib_rcpt_to
 - providers/dime/signet-resolver/dmtplib.c, 1718
- _sgnt_resolv_dmtplib_reset
 - providers/dime/signet-resolver/dmtplib.c, 1718
- _sgnt_resolv_dmtplib_send_and_read
 - providers/dime/signet-resolver/dmtplib.c, 1718
 - providers/dime/signet-resolver/dmtplib.h, 839
- _sgnt_resolv_dmtplib_stats
 - providers/dime/signet-resolver/dmtplib.c, 1718
- _sgnt_resolv_dmtplib_str_to_mode
 - providers/dime/signet-resolver/dmtplib.c, 1719
 - providers/dime/signet-resolver/dmtplib.h, 840
- _sgnt_resolv_dmtplib_verify_signet
 - providers/dime/signet-resolver/dmtplib.c, 1719
- _sgnt_resolv_dmtplib_write_data
 - providers/dime/signet-resolver/dmtplib.c, 1719
 - providers/dime/signet-resolver/dmtplib.h, 840
- _sgnt_resolv_parse_line_code
 - providers/dime/signet-resolver/dmtplib.c, 1720
 - providers/dime/signet-resolver/dmtplib.h, 840
- _sgnt_resolv_read_dmtplib_line
 - providers/dime/signet-resolver/dmtplib.c, 1720
 - providers/dime/signet-resolver/dmtplib.h, 841
- _sgnt_resolv_read_dmtplib_multiline
 - providers/dime/signet-resolver/dmtplib.c, 1720
 - providers/dime/signet-resolver/dmtplib.h, 841
- _sort_rrs_canonical
 - dns.c, 1744

- dns.h, 1758
- _ssl_connect_host
 - ssl.c, 1780
- _ssl_disconnect
 - ssl.c, 1781
- _ssl_fd_loop
 - signet-ssl.h, 1773
 - ssl.c, 1781
- _ssl_get_client_context
 - ssl.c, 1781
- _ssl_initialize
 - ssl.c, 1782
- _ssl_shutdown
 - ssl.c, 1782
- _ssl_starttls
 - ssl.c, 1782
- _str_printf
 - misc.c, 1548
- _t_err_stack
 - error.c, 1523
- _unlink_object
 - providers/dime/signet-resolver/cache.c, 642
 - providers/dime/signet-resolver/cache.h, 661
- _unlock_cache_store
 - providers/dime/signet-resolver/cache.c, 643
 - providers/dime/signet-resolver/cache.h, 661
- _validate_dime_record
 - mrec.c, 1763
- _validate_rrsig_rr
 - dns.c, 1745
- _validate_self_signed
 - signet-ssl.h, 1773
 - ssl.c, 1782
- _verbose
 - misc.c, 1549
 - misc.h, 1554
- _verify_certificate_callback
 - signet-ssl.h, 1773
 - ssl.c, 1783
- _verify_dx_certificate
 - providers/dime/signet-resolver/dmtp.c, 1721
- _verify_ec_sha_signature
 - providers/dime/common/crypto.c, 1205
- _verify_ec_signature
 - providers/dime/common/crypto.c, 1206
- _write_pem_data
 - misc.c, 1549
- A
 - crc.c, 194
- A1
 - crc.c, 194
- abuse
 - magma_t, 91
- accept.c
 - smtp_accept_message, 2145
 - smtp_rollout, 2145

- smtp_store_message, 2146
- smtp_store_spamsig, 2146
- action
 - smtp_inbound_filter_t, 139
- active
 - dmtp_session_t, 54
 - magma_t, 91
- actor
 - dmime_object_t, 48
- add_cached_object
 - cache_pub.c, 1708
- add_cached_object_cmp
 - cache_pub.c, 1708
- add_cached_object_cmp_forced
 - cache_pub.c, 1708
- add_cached_object_forced
 - cache_pub.c, 1708
- address
 - smtp_inbound_prefs_t, 142
 - smtp_recipients_t, 151
 - subnet_t, 156
- address_length_limit
 - magma_t, 91
- addresses.c
 - con_addr, 825
 - con_addr_octet, 825
 - con_addr_presentation, 826
 - con_addr_reversed, 826
 - con_addr_segment, 826
 - con_addr_standard, 827
 - con_addr_subnet, 827
 - con_addr_word, 827
- adler.c
 - hash_adler32, 189
- admin
 - magma_t, 91
- aes.c
 - __attribute__, 1856
 - aes_artifact_decrypt, 1856
 - aes_artifact_encrypt, 1856
 - aes_chunk_decrypt, 1857
 - aes_chunk_encrypt, 1857
 - aes_cipher_key, 1857
 - aes_tag_shard, 1857
 - aes_vector_shard, 1858
 - PRIME_CHUNK_HEAD_LEN, 1855
 - PRIME_CHUNK_KEY_LEN, 1855
 - PRIME_CHUNK_PREFIX_LEN, 1855
 - prime_encrypted_chunk_header_t, 1858
 - prime_encrypted_object_header_t, 1858
 - prime_encrypted_payload_prefix_t, 1858
 - PRIME_OBJECT_HEAD_LEN, 1856
 - PRIME_OBJECT_KEY_LEN, 1856
 - PRIME_OBJECT_PAYLOAD_PREFIX_LEN, 1856
 - PRIME_OBJECT_PREFIX_LEN, 1856
- AES_256_KEK_SIZE
 - dcrypto.h, 1515

- AES_256_KEY_SIZE
 - dcrypto.h, 1515
- AES_256_PADDING_SIZE
 - dcrypto.h, 1515
- aes_artifact_decrypt
 - aes.c, 1856
 - prime/cryptography/cryptography.h, 1382
- aes_artifact_encrypt
 - aes.c, 1856
 - prime/cryptography/cryptography.h, 1382
- AES_BLOCK_LEN
 - prime.h, 1902
- aes_chunk_decrypt
 - aes.c, 1857
 - prime/cryptography/cryptography.h, 1382
- aes_chunk_encrypt
 - aes.c, 1857
 - prime/cryptography/cryptography.h, 1382
- aes_cipher_key
 - aes.c, 1857
 - prime/cryptography/cryptography.h, 1382
- AES_KEY_LEN
 - prime.h, 1902
- AES_TAG_LEN
 - prime.h, 1902
- aes_tag_shard
 - aes.c, 1857
 - prime/cryptography/cryptography.h, 1383
- AES_VECTOR_LEN
 - prime.h, 1902
- aes_vector_shard
 - aes.c, 1858
 - prime/cryptography/cryptography.h, 1383
- agent
 - __attribute__, 18
- alert_alloc
 - alerts.c, 1194
 - objects/meta/meta.h, 911
- ALERT_PRINT
 - misc.h, 1551
- alerts.c
 - alert_alloc, 1194
- algorithm
 - dnskey, 56
 - ds, 59
- alias.c
 - alias_alloc, 1195
- alias_alloc
 - alias.c, 1195
 - objects/meta/meta.h, 911
- ALIGN
 - ed25519-donna-portable.h, 1656
- align
 - align.c, 364
 - memory.h, 378
- align.c
 - align, 364
- ALLOCATED
 - checkers.h, 1262
- allocated
 - secure.c, 390
- allocation.c
 - st_alloc, 485
 - st_alloc_opts, 485
 - st_append_opts, 486
 - st_append_out, 486
 - st_cleanup_variadic, 486
 - st_copy_in, 487
 - st_dupe, 487
 - st_dupe_opts, 487
 - st_free, 488
 - st_import, 489
 - st_import_opts, 490
 - st_merge_opts, 490
 - st_nullify, 490
 - st_output, 491
 - st_realloc, 491
- allocations.h
 - comment, 1259
 - ip_v4_allocations_t, 1259
 - owner, 1259
 - status, 1259
 - weight, 1259
- allow_cross_domain
 - magma_t, 91
- ALTERNATE_PADDING_ALGORITHM_ENABLED
 - dmessage/common.h, 1569
- ALTERNATE_USER_KEY_APPLIED_TO_DATE
 - dmessage/common.h, 1569
- ANSI_COLOR_RED
 - misc.h, 1551
- ANSI_COLOR_RESET
 - misc.h, 1551
- api_endpoint_auth
 - endpoints.c, 2172
 - json_api.h, 2176
- api_endpoint_change_password
 - endpoints.c, 2172
 - json_api.h, 2176
- api_endpoint_delete_user
 - endpoints.c, 2172
 - json_api.h, 2176
- api_endpoint_register
 - endpoints.c, 2172
 - json_api.h, 2176
- api_error
 - helpers.c, 2173
 - json_api.h, 2176
- api_lookup_t, 21
 - callback, 21
 - string, 21
- api_response
 - helpers.c, 2173
 - json_api.h, 2177

- ar_alloc
 - arrays.c, [162](#)
 - buckets.h, [170](#)
- ar_append
 - arrays.c, [162](#)
 - buckets.h, [170](#)
- ar_avail_get
 - arrays.c, [162](#)
 - buckets.h, [170](#)
- ar_dupe
 - arrays.c, [162](#)
 - buckets.h, [170](#)
- ar_field_ar
 - arrays.c, [163](#)
 - buckets.h, [171](#)
- ar_field_ns
 - arrays.c, [163](#)
 - buckets.h, [171](#)
- ar_field_pl
 - arrays.c, [163](#)
 - buckets.h, [171](#)
- ar_field_ptr
 - arrays.c, [164](#)
 - buckets.h, [172](#)
- ar_field_st
 - arrays.c, [164](#)
 - buckets.h, [172](#)
- ar_field_type
 - arrays.c, [164](#)
 - buckets.h, [172](#)
- ar_free
 - arrays.c, [164](#)
 - buckets.h, [172](#)
- ar_length_get
 - arrays.c, [165](#)
 - buckets.h, [173](#)
- ar_length_set
 - arrays.c, [165](#)
 - buckets.h, [173](#)
- arg_name
 - dmtip_argument_t, [51](#)
- arg_name_len
 - dmtip_argument_t, [51](#)
- args
 - dmtip_command_key_t, [52](#)
 - dmtip_command_t, [53](#)
- args.c
 - args_parse, [760](#)
 - cmdline_config_data, [761](#)
 - display_usage, [760](#)
 - exit_and_dump, [761](#)
- args_parse
 - args.c, [760](#)
 - context.h, [763](#)
- ARMORED
 - prime.h, [1904](#)
- armored.h
 - prime_pem_begin, [1920](#)
 - prime_pem_end, [1920](#)
 - prime_pem_unwrap, [1920](#)
 - prime_pem_wrap, [1920](#)
- ARRAY_MAX_ELEMENTS
 - buckets.h, [168](#)
- array_t
 - buckets.h, [169](#)
- ARRAY_TYPE_ARRAY
 - buckets.h, [168](#)
- ARRAY_TYPE_EMPTY
 - buckets.h, [168](#)
- ARRAY_TYPE_NULLER
 - buckets.h, [168](#)
- ARRAY_TYPE_PLACER
 - buckets.h, [168](#)
- ARRAY_TYPE_POINTER
 - buckets.h, [168](#)
- ARRAY_TYPE_SIZER
 - buckets.h, [168](#)
- ARRAY_TYPE_STRINGER
 - buckets.h, [169](#)
- arrays.c
 - ar_alloc, [162](#)
 - ar_append, [162](#)
 - ar_avail_get, [162](#)
 - ar_dupe, [162](#)
 - ar_field_ar, [163](#)
 - ar_field_ns, [163](#)
 - ar_field_pl, [163](#)
 - ar_field_ptr, [164](#)
 - ar_field_st, [164](#)
 - ar_field_type, [164](#)
 - ar_free, [164](#)
 - ar_length_get, [165](#)
 - ar_length_set, [165](#)
- ascii.c
 - chr_alphanumeric, [201](#)
 - chr_ascii, [201](#)
 - chr_blank, [202](#)
 - chr_is_class, [202](#)
 - chr_lower, [202](#)
 - chr_numeric, [202](#)
 - chr_printable, [203](#)
 - chr_punctuation, [203](#)
 - chr_upper, [203](#)
 - chr_whitespace, [203](#)
- ASN1_GENERALIZEDTIME_print_d
 - symbols.h, [2023](#)
- ASN1_INTEGER_to_BN_d
 - symbols.h, [2023](#)
- ASN1_STRING_data_d
 - symbols.c, [1984](#)
 - symbols.h, [2023](#)
- ASN1_STRING_TABLE_cleanup_d
 - symbols.c, [1984](#)
 - symbols.h, [2024](#)

attach
 dmime_message_t, 46
 dmime_object_t, 48
attachment_t
 network/sessions.h, 971
auth.c
 auth_alloc, 1013
 auth_challenge, 1013
 auth_free, 1014
 auth_login, 1014
 auth_response, 1014
auth.h
 AUTH_LOCK_ABUSE, 1017
 AUTH_LOCK_ADMIN, 1017
 AUTH_LOCK_EXPIRED, 1017
 AUTH_LOCK_INACTIVITY, 1017
 AUTH_LOCK_NONE, 1017
 AUTH_LOCK_USER, 1017
 auth_alloc, 1017
 auth_challenge, 1017
 auth_data_fetch, 1018
 auth_data_update_legacy, 1018
 auth_data_update_lock, 1018
 auth_free, 1019
 auth_legacy, 1019
 auth_legacy_alloc, 1019
 auth_legacy_free, 1020
 auth_lock_status_t, 1017
 auth_login, 1020
 auth_response, 1020
 auth_sanitize_address, 1021
 auth_sanitize_username, 1021
 auth_stacie, 1021
 auth_stacie_alloc, 1022
 auth_stacie_cleanup, 1022
 auth_stacie_free, 1023
AUTH_LOCK_ABUSE
 auth.h, 1017
AUTH_LOCK_ADMIN
 auth.h, 1017
AUTH_LOCK_EXPIRED
 auth.h, 1017
AUTH_LOCK_INACTIVITY
 auth.h, 1017
AUTH_LOCK_NONE
 auth.h, 1017
AUTH_LOCK_USER
 auth.h, 1017
auth_alloc
 auth.c, 1013
 auth.h, 1017
auth_challenge
 auth.c, 1013
 auth.h, 1017
auth_data_fetch
 auth.h, 1018
 objects/auth/datatier.c, 683
auth_data_update_legacy
 auth.h, 1018
 objects/auth/datatier.c, 683
auth_data_update_lock
 auth.h, 1018
 objects/auth/datatier.c, 684
auth_free
 auth.c, 1014
 auth.h, 1019
AUTH_GET_BY_ADDRESS
 queries.h, 2071
AUTH_GET_BY_USERID
 queries.h, 2071
auth_keyslot
 dmime_chunk_key_t, 41
auth_legacy
 auth.h, 1019
 legacy.c, 1024
auth_legacy_alloc
 auth.h, 1019
 legacy.c, 1024
auth_legacy_free
 auth.h, 1020
 legacy.c, 1024
auth_legacy_t, 22
 key, 22
 token, 22
auth_lock_status_t
 auth.h, 1017
auth_login
 auth.c, 1014
 auth.h, 1020
auth_recip
 dmime_envelope_object_t, 44
auth_recip_fp
 dmime_envelope_object_t, 44
auth_response
 auth.c, 1014
 auth.h, 1020
auth_sanitize_address
 auth.h, 1021
 username.c, 1028
auth_sanitize_username
 auth.h, 1021
 username.c, 1028
auth_stacie
 auth.h, 1021
 stacie.c, 1025
auth_stacie_alloc
 auth.h, 1022
 stacie.c, 1025
auth_stacie_cleanup
 auth.h, 1022
 stacie.c, 1026
auth_stacie_free
 auth.h, 1023
 stacie.c, 1026

- auth_stacie_t, 23
 - ephemeral, 23
 - keys, 23
 - master, 23
 - password, 23
 - tokens, 23
 - verification, 23
- auth_t, 25
 - bonus, 25
 - ephemeral, 25
 - key, 25
 - keys, 26
 - legacy, 26
 - locked, 26
 - master, 26
 - nonce, 26
 - salt, 26
 - seasoning, 26
 - status, 26
 - token, 26
 - tokens, 26
 - username, 27
 - usernum, 27
 - verification, 27
- AUTH_UPDATE_LEGACY_TO_STACIE
 - queries.h, 2071
- AUTH_UPDATE_USER_LOCK
 - queries.h, 2071
- authenticated
 - smtp_session_t, 152
- author
 - dmime_object_t, 48
- author_full_sig
 - dmime_message_t, 46
- author_tree_sig
 - dmime_message_t, 46
- autoreply
 - smtp_inbound_prefs_t, 142
- available
 - magma_t, 91
- B
 - crc.c, 195
- B64
 - signet/common.h, 1576
- B64_DECODED_LEN
 - misc.h, 1551
- B64_ENCODED_LEN
 - misc.h, 1551
- b64decode
 - misc_pub.c, 1556
- b64decode_nopad
 - misc_pub.c, 1556
- b64encode
 - misc_pub.c, 1556
- b64encode_nopad
 - misc_pub.c, 1557
- backtrace.c
 - backtrace_print, 258
- backtrace_print
 - backtrace.c, 258
 - host.h, 288
- bad_command_cutoff
 - server_config_t, 127
- bad_command_delay
 - server_config_t, 127
- banner
 - server_config_t, 127
- base64
 - mappings_t, 108
- base64.c
 - base64_decode, 232
 - base64_decode_mod, 233
 - base64_decode_opts, 233
 - base64_decoded_length, 233
 - base64_decoded_length_mod, 233
 - base64_encode, 234
 - base64_encode_mod, 234
 - base64_encode_opts, 234
 - base64_encode_wrap, 235
 - base64_encoded_length, 235
 - base64_encoded_length_mod, 235
 - base64_encoded_length_wrap, 235
- BASE64_LINE_WRAP_CRLF
 - encodings.h, 239
- BASE64_LINE_WRAP_LF
 - encodings.h, 239
- BASE64_LINE_WRAP_NONE
 - encodings.h, 239
- base64_decode
 - base64.c, 232
 - encodings.h, 239
- base64_decode_mod
 - base64.c, 233
 - encodings.h, 239
- base64_decode_opts
 - base64.c, 233
 - encodings.h, 240
- base64_decoded_length
 - base64.c, 233
 - encodings.h, 240
- base64_decoded_length_mod
 - base64.c, 233
 - encodings.h, 240
- base64_encode
 - base64.c, 234
 - encodings.h, 240
- base64_encode_mod
 - base64.c, 234
 - encodings.h, 241
- base64_encode_opts
 - base64.c, 234
 - encodings.h, 241
- base64_encode_wrap

- base64.c, 235
- encodings.h, 242
- base64_encoded_length
 - base64.c, 235
 - encodings.h, 242
- base64_encoded_length_mod
 - base64.c, 235
 - encodings.h, 242
- base64_encoded_length_wrap
 - base64.c, 235
 - encodings.h, 242
- BASE64_LINE_WRAP_LENGTH
 - encodings.h, 239
- base64_mod
 - mappings_t, 108
- base64_wrap_t
 - encodings.h, 239
- batch_no_errors
 - test.c, 1689
- batch_wrong_message
 - test.c, 1689
- batch_wrong_pk
 - test.c, 1689
- batch_wrong_sig
 - test.c, 1689
- batch_heap
 - ed25519-donna-batchverify.h, 1651
- batch_heap_t, 28
 - heap, 28
 - points, 28
 - r, 28
 - scalars, 28
 - size, 28
- batch_point_buffer
 - ed25519-donna-batchverify.h, 1652
 - test.c, 1689
- batch_test
 - test.c, 1689
- batch_test_t
 - test.c, 1689
- bignum25519
 - curve25519-donna-32bit.h, 1638
 - curve25519-donna-64bit.h, 1640
 - curve25519-donna-sse2.h, 1643
- bignum25519align16
 - curve25519-donna-32bit.h, 1638
- bignum256modm
 - modm-donna-32bit.h, 1683
 - modm-donna-64bit.h, 1684
- bignum256modm_bits_per_limb
 - modm-donna-32bit.h, 1683
 - modm-donna-64bit.h, 1684
- bignum256modm_element_t
 - modm-donna-32bit.h, 1683
 - modm-donna-64bit.h, 1684
- bignum256modm_limb_size
 - modm-donna-32bit.h, 1683
- modm-donna-64bit.h, 1684
- bin
 - media_type_t, 109
- BINARY
 - prime.h, 1904
- binary
 - multi_t, 111
- binary.h
 - __attribute__, 1925
 - prime_field_data_t, 1925
 - prime_field_get, 1925
 - prime_field_size_length, 1926
 - prime_field_size_max, 1926
 - prime_field_t, 1931
 - prime_field_type_t, 1925
 - prime_field_write, 1926
 - prime_header_encrypted_message_write, 1926
 - prime_header_encrypted_org_key_write, 1926
 - prime_header_encrypted_user_key_write, 1926
 - prime_header_length, 1927
 - prime_header_org_key_write, 1927
 - prime_header_org_signet_write, 1927
 - prime_header_read, 1927
 - prime_header_user_key_write, 1928
 - prime_header_user_signet_write, 1928
 - prime_header_user_signing_request_write, 1928
 - prime_header_write, 1928
 - prime_object_alloc, 1928
 - prime_object_free, 1929
 - prime_object_size_max, 1929
 - prime_object_size_min, 1929
 - prime_object_t, 1931
 - prime_object_type, 1929
 - prime_reader_open, 1929
 - prime_reader_payload, 1929
 - prime_reader_size, 1930
 - prime_reader_t, 1931
 - prime_reader_type, 1930
 - prime_size_t, 1925
 - prime_unpack, 1930
 - prime_unpack_fields, 1930
 - prime_unpack_validate, 1931
- BIO_free_all_d
 - symbols.c, 1985
 - symbols.h, 2024
- BIO_free_d
 - symbols.c, 1985
 - symbols.h, 2024
- BIO_new_fp_d
 - symbols.c, 1985
 - symbols.h, 2024
- BIO_new_socket_d
 - symbols.c, 1985
 - symbols.h, 2024
- BIO_sock_cleanup_d
 - symbols.c, 1985
 - symbols.h, 2024

- BIO_vprintf_d
 - symbols.h, 2024
- bitwise_and
 - memory.h, 378
 - memory/bitwise.c, 365
- bitwise_count
 - memory.h, 378
 - memory/bitwise.c, 365
- bitwise_or
 - memory.h, 378
 - memory/bitwise.c, 365
- bitwise_xor
 - memory.h, 378
 - memory/bitwise.c, 365
- bl
 - multi_t, 111
- blacklists
 - magma_t, 91
- BLOCK
 - strings.h, 539
- block
 - tok_state_t, 160
- BLOCK_T
 - strings.h, 543
- block_t
 - strings.h, 577
- BLOCKBUF
 - strings.h, 539
- BN_bin2bn_d
 - symbols.h, 2024
- BN_bn2bin_d
 - symbols.c, 1985
 - symbols.h, 2024
- BN_bn2dec_d
 - symbols.c, 1985
 - symbols.h, 2024
- BN_bn2hex_d
 - symbols.c, 1985
 - symbols.h, 2024
- BN_bn2mpi_d
 - symbols.c, 1985
 - symbols.h, 2024
- BN_cmp_d
 - symbols.c, 1985
 - symbols.h, 2025
- BN_CTX_free_d
 - symbols.c, 1985
 - symbols.h, 2025
- BN_CTX_new_d
 - symbols.c, 1985
 - symbols.h, 2025
- BN_CTX_start_d
 - symbols.c, 1985
 - symbols.h, 2025
- BN_free_d
 - symbols.c, 1985
 - symbols.h, 2025
- BN_hex2bn_d
 - symbols.c, 1986
 - symbols.h, 2025
- BN_mpi2bn_d
 - symbols.c, 1986
 - symbols.h, 2025
- BN_num_bits_d
 - symbols.c, 1986
 - symbols.h, 2025
- BN_num_bytes_d
 - cryptography/cryptography.h, 1348
- body
 - __attribute__, 18
 - imap_fetch_dataitems_t, 78
 - mail_mime_t, 106
- bodystructure
 - imap_fetch_dataitems_t, 78
- bonus
 - auth_t, 25
- bool_t
 - core.h, 229
- bounces
 - smtp_inbound_prefs_t, 142
- boundary
 - mail_mime_t, 106
- bracket.c
 - bracket_extract_pl, 471
- bracket_extract_pl
 - bracket.c, 471
 - special.h, 472
- buckets.h
 - __attribute__, 169
 - ar_alloc, 170
 - ar_append, 170
 - ar_avail_get, 170
 - ar_dupe, 170
 - ar_field_ar, 171
 - ar_field_ns, 171
 - ar_field_pl, 171
 - ar_field_ptr, 172
 - ar_field_st, 172
 - ar_field_type, 172
 - ar_free, 172
 - ar_length_get, 173
 - ar_length_set, 173
 - ARRAY_MAX_ELEMENTS, 168
 - array_t, 169
 - ARRAY_TYPE_ARRAY, 168
 - ARRAY_TYPE_EMPTY, 168
 - ARRAY_TYPE_NULLER, 168
 - ARRAY_TYPE_PLACER, 168
 - ARRAY_TYPE_POINTER, 168
 - ARRAY_TYPE_SIZER, 168
 - ARRAY_TYPE_STRINGER, 169
 - M_POOL_STATUS, 169
 - MAGMA_CORE_POOL_OBJECTS_LIMIT, 169
 - MAGMA_CORE_POOL_TIMEOUT_LIMIT, 169

- PL_AVAILABLE, 169
- PL_ERROR, 169
- PL_RESERVED, 169
- pool_alloc, 173
- pool_free, 174
- pool_get_available, 174
- pool_get_count, 174
- pool_get_failures, 175
- pool_get_obj, 175
- pool_get_status, 175
- pool_get_timeout, 176
- pool_pull, 176
- pool_release, 176
- pool_set_obj, 177
- pool_set_status, 177
- pool_swap_obj, 178
- pool_t, 179
- stacker_alloc, 178
- stacker_free, 178
- stacker_node_t, 179
- stacker_nodes, 178
- stacker_pop, 179
- stacker_push, 179
- stacker_t, 180
- status_t, 169
- BUF_strlcat_d
 - symbols.c, 1986
 - symbols.h, 2025
- buflen
 - magma.h, 823
- bufptr
 - magma.h, 823
- build.c
 - build_commit, 795
 - build_stamp, 795
 - build_version, 795
 - build_version_major, 796
 - build_version_minor, 796
 - build_version_patch, 796
- build_commit
 - build.c, 795
 - status.h, 812
- build_stamp
 - build.c, 795
 - status.h, 812
- build_version
 - build.c, 795
 - status.h, 812
- build_version_major
 - build.c, 796
 - status.h, 812
- build_version_minor
 - build.c, 796
 - status.h, 812
- build_version_patch
 - build.c, 796
 - status.h, 812
- bypass
 - smtp_session_t, 152
- bypass_addr
 - magma_t, 91
- bypass_subnets
 - magma_t, 92
- byte_t
 - core.h, 229
- bytes
 - secure.c, 390
- bytes_data_size
 - signet_field_key_t, 134
- bytes_name_size
 - signet_field_key_t, 134
- BZ2_bzBuffToBuffCompress_d
 - symbols.c, 1986
 - symbols.h, 2025
- BZ2_bzBuffToBuffDecompress_d
 - symbols.c, 1986
 - symbols.h, 2025
- BZ2_bzlibVersion_d
 - symbols.c, 1986
 - symbols.h, 2025
- bzip.c
 - bzip_version, 1287
 - compress_bzip, 1286
 - decompress_bzip, 1286
 - lib_load_bzip, 1287
 - lib_version_bzip, 1287
- bzip_version
 - bzip.c, 1287
- C
 - crc.c, 195
- CA_DIR
 - signet-ssl.h, 1770
- CA_FILE
 - signet-ssl.h, 1770
- cache
 - magma_t, 92
- cache_add
 - consumers.h, 1309
 - providers/consumers/cache.c, 619
- cache_alloc
 - engine/config/cache/cache.c, 611
 - engine/config/cache/cache.h, 644
- cache_append
 - consumers.h, 1309
 - providers/consumers/cache.c, 619
- cache_config
 - engine/config/cache/cache.c, 611
 - engine/config/cache/cache.h, 644
- cache_decrement
 - consumers.h, 1309
 - providers/consumers/cache.c, 619
- cache_delete
 - consumers.h, 1310

- providers/consumers/cache.c, 620
- cache_flush
 - consumers.h, 1310
 - providers/consumers/cache.c, 620
- cache_free
 - engine/config/cache/cache.c, 612
 - engine/config/cache/cache.h, 645
- cache_get
 - consumers.h, 1310
 - providers/consumers/cache.c, 620
- cache_get_u64
 - consumers.h, 1310
 - providers/consumers/cache.c, 620
- cache_increment
 - consumers.h, 1311
 - providers/consumers/cache.c, 621
- cache_keys
 - engine/config/cache/keys.h, 663
- cache_keys_t, 29
 - description, 29
 - name, 29
 - norm, 29
 - offset, 29
 - required, 29
- cache_output_help
 - engine/config/cache/cache.c, 612
 - engine/config/cache/cache.h, 645
- cache_output_settings
 - engine/config/cache/cache.c, 612
 - engine/config/cache/cache.h, 645
- CACHE_PERM_ADD
 - providers/dime/signet-resolver/cache.h, 651
- CACHE_PERM_ALL_FLAGS
 - providers/dime/signet-resolver/cache.h, 651
- CACHE_PERM_DEFAULT
 - providers/dime/signet-resolver/cache.h, 651
- CACHE_PERM_DELETE
 - providers/dime/signet-resolver/cache.h, 651
- CACHE_PERM_LOAD
 - providers/dime/signet-resolver/cache.h, 651
- CACHE_PERM_NONE
 - providers/dime/signet-resolver/cache.h, 651
- CACHE_PERM_READ
 - providers/dime/signet-resolver/cache.h, 651
- CACHE_PERM_SAVE
 - providers/dime/signet-resolver/cache.h, 651
- cache_pool
 - providers/consumers/cache.c, 623
- cache_pub.c
 - add_cached_object, 1708
 - add_cached_object_cmp, 1708
 - add_cached_object_cmp_forced, 1708
 - add_cached_object_forced, 1708
 - cached_object_exists, 1709
 - cached_object_exists_cmp, 1709
 - destroy_cache_entry, 1709
 - find_cached_object, 1709
 - find_cached_object_cmp, 1709
 - get_cache_location, 1709
 - get_cache_obj_data, 1709
 - get_dime_dir_location, 1709
 - load_cache_contents, 1709
 - remove_cached_object, 1710
 - remove_cached_object_cmp, 1710
 - save_cache_contents, 1710
 - set_cache_location, 1710
 - set_cache_permissions, 1710
- cache_set
 - consumers.h, 1311
 - providers/consumers/cache.c, 621
- cache_set_u64
 - consumers.h, 1312
 - providers/consumers/cache.c, 622
- cache_set_value
 - engine/config/cache/cache.c, 613
 - engine/config/cache/cache.h, 646
- cache_silent_add
 - consumers.h, 1312
 - providers/consumers/cache.c, 622
- cache_start
 - consumers.h, 1312
 - providers/consumers/cache.c, 622
- cache_stop
 - consumers.h, 1312
 - providers/consumers/cache.c, 622
- cache_t, 30
 - name, 30
 - port, 30
 - weight, 30
- cache_validate
 - engine/config/cache/cache.c, 613
 - engine/config/cache/cache.h, 646
- cached_data_dnskey
 - providers/dime/signet-resolver/cache.h, 652
- cached_data_drec
 - providers/dime/signet-resolver/cache.h, 652
- cached_data_ds
 - providers/dime/signet-resolver/cache.h, 652
- cached_data_ocsp
 - providers/dime/signet-resolver/cache.h, 652
- cached_data_signet
 - providers/dime/signet-resolver/cache.h, 652
- cached_data_unknown
 - providers/dime/signet-resolver/cache.h, 652
- cached_data_type_t
 - providers/dime/signet-resolver/cache.h, 652
- cached_object, 31
 - data, 31
 - dtype, 31
 - expiration, 32
 - id, 32
 - next, 32
 - persists, 32
 - prev, 32

- relaxed, 32
- shadow, 33
- timestamp, 33
- ttl, 33
- cached_object_exists
 - cache_pub.c, 1709
- cached_object_exists_cmp
 - cache_pub.c, 1709
- cached_object_t
 - providers/dime/signet-resolver/cache.h, 651
- cached_store_comparator_t
 - providers/dime/signet-resolver/cache.h, 651
- cached_store_t, 34
 - clone, 34
 - description, 34
 - deserialize, 34
 - destructor, 35
 - dtype, 35
 - dump, 35
 - head, 35
 - internal, 35
 - lock, 35
 - serialize, 35
- cached_stores
 - providers/dime/signet-resolver/cache.c, 643
 - providers/dime/signet-resolver/cache.h, 662
- callback
 - api_lookup_t, 21
- captcha.c
 - register_captcha_generate, 2238
 - register_captcha_random_font, 2238
 - register_captcha_write_noise, 2238
- carry_pass
 - curve25519-donna-32bit.h, 1637
 - curve25519-donna-sse2.h, 1642
- carry_pass_final
 - curve25519-donna-32bit.h, 1637
 - curve25519-donna-sse2.h, 1642
- carry_pass_full
 - curve25519-donna-32bit.h, 1637
 - curve25519-donna-sse2.h, 1642
- case.c
 - lower_chr, 392
 - lower_st, 392
 - upper_chr, 392
 - upper_st, 393
- certificate
 - server_config_t, 127
 - server_t, 130
- characters
 - mappings_t, 108
- checked
 - smtp_session_t, 152
- checkers.c
 - smtp_add_bypass_entry, 2147
 - smtp_bypass_check, 2147
 - smtp_check_filters, 2148
 - smtp_check_greylist, 2148
 - smtp_check_rbl, 2148
- checkers.h
 - ALLOCATED, 1262
 - DIRECT, 1262
 - dkim_memory_alloc, 1262
 - dkim_memory_free, 1262
 - dkim_signature_create, 1263
 - dkim_signature_verify, 1263
 - dkim_start, 1264
 - dkim_stop, 1264
 - DOMESTIC, 1262
 - dspam_check, 1264
 - dspam_start, 1264
 - dspam_stop, 1265
 - dspam_train, 1265
 - FOREIGN, 1262
 - IP_RANDOMIZER_POOL, 1261
 - IP_RANDOMIZER_PUSH_MAX, 1261
 - IP_RANDOMIZER_PUSH_MIN, 1261
 - lib_load_clamav, 1265
 - lib_load_dkim, 1265
 - lib_load_dspam, 1266
 - lib_load_spf, 1266
 - lib_version_clamav, 1266
 - lib_version_dkim, 1266
 - lib_version_dspam, 1267
 - lib_version_spf, 1267
 - REGISTRY, 1262
 - RESERVED, 1262
 - spf_check, 1267
 - spf_start, 1267
 - spf_stop, 1268
 - UNALLOCATED, 1262
 - virus_check, 1268
 - virus_engine_create, 1268
 - virus_engine_destroy, 1269
 - virus_engine_refresh, 1269
 - virus_sigs_loaded, 1269
 - virus_sigs_total, 1269
 - virus_start, 1270
 - virus_stop, 1270
- checksum.h
 - crc24_checksum, 190
 - crc24_final, 191
 - crc24_init, 191
 - crc24_update, 191
 - crc32_checksum, 191
 - crc32_update, 191
 - crc64_checksum, 192
 - crc64_update, 192
 - hash_adler32, 192
 - hash_fletcher32, 192
 - hash_murmur32, 193
 - hash_murmur64, 193
- children
 - mail_mime_t, 106

- chr_alphanumeric
 - ascii.c, [201](#)
 - classify.h, [205](#)
- chr_ascii
 - ascii.c, [201](#)
 - classify.h, [205](#)
- chr_blank
 - ascii.c, [202](#)
 - classify.h, [206](#)
- chr_is_class
 - ascii.c, [202](#)
 - classify.h, [206](#)
- chr_isprint
 - misc.h, [1552](#)
- chr_isspace
 - misc.h, [1552](#)
- chr_lower
 - ascii.c, [202](#)
 - classify.h, [206](#)
- chr_numeric
 - ascii.c, [202](#)
 - classify.h, [206](#)
- chr_printable
 - ascii.c, [203](#)
 - classify.h, [207](#)
- chr_punctuation
 - ascii.c, [203](#)
 - classify.h, [207](#)
- chr_t
 - core.h, [229](#)
- chr_upper
 - ascii.c, [203](#)
 - classify.h, [207](#)
- chr_whitespace
 - ascii.c, [203](#)
 - classify.h, [207](#)
- CHUNK_SECTION_ATTACH
 - dmessage/common.h, [1571](#)
- CHUNK_SECTION_DISPLAY
 - dmessage/common.h, [1571](#)
- CHUNK_SECTION_ENVELOPE
 - dmessage/common.h, [1571](#)
- CHUNK_SECTION_METADATA
 - dmessage/common.h, [1571](#)
- CHUNK_SECTION_NONE
 - dmessage/common.h, [1571](#)
- CHUNK_SECTION_SIG
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_ALTERNATE
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_ATTACH_CONTENT
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_DESTINATION
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_DISPLAY_CONTENT
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_EPHEMERAL
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_META_COMMON
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_META_OTHER
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_NONE
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_ORIGIN
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_SIG_AUTHOR_FULL
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_SIG_AUTHOR_TREE
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_SIG_ORIGIN_DISPLAY_BOUNCE
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_SIG_ORIGIN_FULL
 - dmessage/common.h, [1571](#)
- CHUNK_TYPE_SIG_ORIGIN_META_BOUNCE
 - dmessage/common.h, [1571](#)
- chunk_buffer_read
 - chunks.c, [1871](#)
 - chunks.h, [1874](#)
- chunk_buffer_size
 - chunks.c, [1871](#)
 - chunks.h, [1874](#)
- chunk_header_read
 - chunks.c, [1871](#)
 - chunks.h, [1874](#)
- CHUNK_HEADER_SIZE
 - dmessage/common.h, [1570](#)
- chunk_header_size
 - chunks.c, [1871](#)
 - chunks.h, [1875](#)
- chunk_header_type
 - chunks.c, [1871](#)
 - chunks.h, [1875](#)
- chunk_header_write
 - chunks.c, [1872](#)
 - chunks.h, [1875](#)
- CHUNK_LENGTH_SIZE
 - dmessage/common.h, [1570](#)
- chunks.c
 - chunk_buffer_read, [1871](#)
 - chunk_buffer_size, [1871](#)
 - chunk_header_read, [1871](#)
 - chunk_header_size, [1871](#)
 - chunk_header_type, [1871](#)
 - chunk_header_write, [1872](#)
- chunks.h
 - chunk_buffer_read, [1874](#)
 - chunk_buffer_size, [1874](#)
 - chunk_header_read, [1874](#)
 - chunk_header_size, [1875](#)
 - chunk_header_type, [1875](#)
 - chunk_header_write, [1875](#)
 - encrypted_chunk_alloc, [1875](#)
 - encrypted_chunk_buffer, [1875](#)

- encrypted_chunk_cleanup, [1875](#)
- encrypted_chunk_free, [1876](#)
- encrypted_chunk_get, [1876](#)
- encrypted_chunk_set, [1876](#)
- ephemeral_chunk_alloc, [1876](#)
- ephemeral_chunk_buffer, [1876](#)
- ephemeral_chunk_cleanup, [1876](#)
- ephemeral_chunk_free, [1877](#)
- ephemeral_chunk_get, [1877](#)
- ephemeral_chunk_set, [1877](#)
- keks_alloc, [1877](#)
- keks_cleanup, [1877](#)
- keks_free, [1877](#)
- keks_get, [1878](#)
- keks_set, [1878](#)
- signature_full_get, [1878](#)
- signature_full_verify, [1878](#)
- signature_tree_add, [1878](#)
- signature_tree_alloc, [1879](#)
- signature_tree_cleanup, [1879](#)
- signature_tree_free, [1879](#)
- signature_tree_get, [1879](#)
- signature_tree_verify, [1879](#)
- slots_actors, [1879](#)
- slots_alloc, [1880](#)
- slots_buffer, [1880](#)
- slots_cleanup, [1880](#)
- slots_count, [1880](#)
- slots_free, [1880](#)
- slots_get, [1880](#)
- slots_key, [1881](#)
- slots_set, [1881](#)
- cipher_block_length
 - ciphers.c, [1331](#)
 - cryptography/cryptography.h, [1350](#)
- cipher_id
 - ciphers.c, [1331](#)
 - cryptography/cryptography.h, [1350](#)
- cipher_key_length
 - ciphers.c, [1332](#)
 - cryptography/cryptography.h, [1350](#)
- cipher_name
 - ciphers.c, [1332](#)
 - cryptography/cryptography.h, [1351](#)
- cipher_numeric_id
 - ciphers.c, [1332](#)
 - cryptography/cryptography.h, [1351](#)
- cipher_t
 - cryptography/cryptography.h, [1349](#)
- cipher_vector_length
 - ciphers.c, [1332](#)
 - cryptography/cryptography.h, [1351](#)
- ciphers.c
 - cipher_block_length, [1331](#)
 - cipher_id, [1331](#)
 - cipher_key_length, [1332](#)
 - cipher_name, [1332](#)
 - cipher_numeric_id, [1332](#)
 - cipher_vector_length, [1332](#)
 - CKEY_EMPTY
 - dmsg.c, [1591](#)
 - cl_countsigs_d
 - symbols.c, [1986](#)
 - symbols.h, [2025](#)
 - cl_engine_compile_d
 - symbols.c, [1986](#)
 - symbols.h, [2026](#)
 - cl_engine_free_d
 - symbols.c, [1986](#)
 - symbols.h, [2026](#)
 - cl_engine_new_d
 - symbols.c, [1986](#)
 - symbols.h, [2026](#)
 - cl_engine_set_num_d
 - symbols.c, [1986](#)
 - symbols.h, [2026](#)
 - cl_engine_set_str_d
 - symbols.c, [1987](#)
 - symbols.h, [2026](#)
 - cl_init_d
 - symbols.c, [1987](#)
 - symbols.h, [2026](#)
 - cl_load_d
 - symbols.c, [1987](#)
 - symbols.h, [2026](#)
 - cl_retver_d
 - symbols.c, [1987](#)
 - symbols.h, [2026](#)
 - cl_scandesc_d
 - symbols.c, [1987](#)
 - symbols.h, [2026](#)
 - cl_shutdown_d
 - symbols.c, [1987](#)
 - symbols.h, [2026](#)
 - cl_statchkdir_d
 - symbols.c, [1987](#)
 - symbols.h, [2026](#)
 - cl_statfree_d
 - symbols.c, [1987](#)
 - symbols.h, [2026](#)
 - cl_statinidir_d
 - symbols.c, [1987](#)
 - symbols.h, [2027](#)
 - cl_strerror_d
 - symbols.c, [1987](#)
 - symbols.h, [2027](#)
 - clamav.c
 - lib_load_clamav, [1271](#)
 - lib_version_clamav, [1271](#)
 - virus_check, [1272](#)
 - virus_engine, [1274](#)
 - virus_engine_create, [1272](#)
 - virus_engine_destroy, [1272](#)
 - virus_engine_refresh, [1273](#)

- virus_lock, 1274
- virus_sigs, 1274
- virus_sigs_loaded, 1273
- virus_sigs_total, 1273
- virus_spool, 1274
- virus_start, 1273
- virus_stat, 1274
- virus_stop, 1274
- clamp.c
 - int16_clamp, 400
 - int32_clamp, 400
 - int64_clamp, 401
 - int8_clamp, 401
 - uint16_clamp, 401
 - uint32_clamp, 402
 - uint64_clamp, 402
 - uint8_clamp, 402
- classify.h
 - chr_alphanumeric, 205
 - chr_ascii, 205
 - chr_blank, 206
 - chr_is_class, 206
 - chr_lower, 206
 - chr_numeric, 206
 - chr_printable, 207
 - chr_punctuation, 207
 - chr_upper, 207
 - chr_whitespace, 207
- cleanup.c
 - mail_destroy_header, 1097
 - mail_message_cleanup, 1097
- client_close
 - clients.c, 829
 - network/network.h, 934
- client_connect
 - clients.c, 829
 - network/network.h, 934
- client_print
 - network/network.h, 935
 - write.c, 1009
- client_read
 - network/network.h, 935
 - read.c, 967
- client_read_line
 - network/network.h, 935
 - read.c, 967
- client_secure
 - clients.c, 829
 - network/network.h, 936
- client_status
 - clients.c, 830
 - network/network.h, 936
- client_t
 - network/network.h, 950
- client_write
 - network/network.h, 936
 - write.c, 1009
- clients.c
 - client_close, 829
 - client_connect, 829
 - client_secure, 829
 - client_status, 830
- clone
 - cached_store_t, 34
- close
 - magma_t, 92
- cmdline_config_data
 - args.c, 761
 - global.c, 727
- cmp_mt_mt
 - multi.c, 510
 - strings.h, 543
- code
 - derror_t, 36
 - log_level_t, 85
- color.c
 - color_blue, 259
 - color_blue_bold, 259
 - color_blue_intense, 260
 - color_blue_intense_bold, 260
 - color_blue_underline, 260
 - color_cyan, 260
 - color_cyan_bold, 260
 - color_cyan_intense, 260
 - color_cyan_intense_bold, 260
 - color_cyan_underline, 260
 - color_green, 260
 - color_green_bold, 261
 - color_green_intense, 261
 - color_green_intense_bold, 261
 - color_green_underline, 261
 - color_purple, 261
 - color_purple_bold, 261
 - color_purple_intense, 261
 - color_purple_intense_bold, 261
 - color_purple_underline, 261
 - color_red, 262
 - color_red_bold, 262
 - color_red_intense, 262
 - color_red_intense_bold, 262
 - color_red_underline, 262
 - color_reset, 262
 - color_supported, 262
 - color_white, 262
 - color_white_bold, 263
 - color_white_intense, 263
 - color_white_intense_bold, 263
 - color_white_underline, 263
 - color_yellow, 263
 - color_yellow_bold, 263
 - color_yellow_intense, 263
 - color_yellow_intense_bold, 263
 - color_yellow_underline, 263
- color_blue

color.c, [259](#)
host.h, [288](#)
color_blue_bold
color.c, [259](#)
host.h, [288](#)
color_blue_intense
color.c, [260](#)
host.h, [288](#)
color_blue_intense_bold
color.c, [260](#)
host.h, [288](#)
color_blue_underline
color.c, [260](#)
host.h, [288](#)
color_cyan
color.c, [260](#)
host.h, [288](#)
color_cyan_bold
color.c, [260](#)
host.h, [288](#)
color_cyan_intense
color.c, [260](#)
host.h, [289](#)
color_cyan_intense_bold
color.c, [260](#)
host.h, [289](#)
color_cyan_underline
color.c, [260](#)
host.h, [289](#)
color_green
color.c, [260](#)
host.h, [289](#)
color_green_bold
color.c, [261](#)
host.h, [289](#)
color_green_intense
color.c, [261](#)
host.h, [289](#)
color_green_intense_bold
color.c, [261](#)
host.h, [289](#)
color_green_underline
color.c, [261](#)
host.h, [289](#)
color_purple
color.c, [261](#)
host.h, [289](#)
color_purple_bold
color.c, [261](#)
host.h, [290](#)
color_purple_intense
color.c, [261](#)
host.h, [290](#)
color_purple_intense_bold
color.c, [261](#)
host.h, [290](#)
color_purple_underline
color.c, [261](#)
host.h, [290](#)
color_red
color.c, [262](#)
host.h, [290](#)
color_red_bold
color.c, [262](#)
host.h, [290](#)
color_red_intense
color.c, [262](#)
host.h, [290](#)
color_red_intense_bold
color.c, [262](#)
host.h, [290](#)
color_red_underline
color.c, [262](#)
host.h, [290](#)
color_reset
color.c, [262](#)
host.h, [291](#)
color_supported
color.c, [262](#)
host.h, [291](#)
color_white
color.c, [262](#)
host.h, [291](#)
color_white_bold
color.c, [263](#)
host.h, [291](#)
color_white_intense
color.c, [263](#)
host.h, [291](#)
color_white_intense_bold
color.c, [263](#)
host.h, [291](#)
color_white_underline
color.c, [263](#)
host.h, [291](#)
color_yellow
color.c, [263](#)
host.h, [291](#)
color_yellow_bold
color.c, [263](#)
host.h, [292](#)
color_yellow_intense
color.c, [263](#)
host.h, [292](#)
color_yellow_intense_bold
color.c, [263](#)
host.h, [292](#)
color_yellow_underline
color.c, [263](#)
host.h, [292](#)
com_name
dmtp_command_key_t, [52](#)
com_name_len
dmtp_command_key_t, [52](#)

- command
 - transaction.c, 1493
- command_t
 - network/network.h, 950
- comment
 - allocations.h, 1259
- common
 - queries.h, 2082
- common_headers
 - dmime_message_t, 46
 - dmime_object_t, 48
- COMP_zlib_cleanup_d
 - symbols.c, 1987
 - symbols.h, 2027
- compare.h
 - mm_cmp_ci_eq, 209
 - mm_cmp_cs_eq, 210
 - st_cmp_ci_ends, 210
 - st_cmp_ci_eq, 210
 - st_cmp_ci_starts, 211
 - st_cmp_cs_ends, 211
 - st_cmp_cs_eq, 212
 - st_cmp_cs_starts, 212
 - st_search_chr, 212
 - st_search_ci, 213
 - st_search_cs, 213
- completed
 - teacher_data_t, 157
- composition_t
 - network/sessions.h, 971
- compress.c
 - compress_alloc, 1288
 - compress_block_length, 1289
 - compress_body_data, 1289
 - compress_body_hash, 1289
 - compress_body_length, 1289
 - compress_body_offset, 1290
 - compress_cleanup, 1290
 - compress_free, 1290
 - compress_import, 1290
 - compress_orig_hash, 1291
 - compress_orig_length, 1291
 - compress_total_length, 1291
- compress.h
 - __attribute__, 1294
 - COMPRESS_ENGINE_BZIP, 1294
 - COMPRESS_ENGINE_LZO, 1294
 - COMPRESS_ENGINE_ZLIB, 1294
 - compress_alloc, 1294
 - compress_block_length, 1294
 - compress_body_data, 1294
 - compress_body_hash, 1295
 - compress_body_length, 1295
 - compress_body_offset, 1295
 - compress_bzip, 1295
 - compress_cleanup, 1296
 - COMPRESS_ENGINE
 - compress.h, 1301
 - compress_free
 - compress.c, 1290
 - compress_free, 1296
 - compress_head_t, 1301
 - compress_import, 1296
 - compress_lzo, 1297
 - compress_orig_hash, 1297
 - compress_orig_length, 1297
 - compress_t, 1293
 - compress_total_length, 1297
 - compress_zlib, 1298
 - decompress_block_lzo, 1298
 - decompress_bzip, 1298
 - decompress_lzo, 1299
 - decompress_zlib, 1299
 - engine_compress, 1299
 - engine_decompress, 1299
 - lib_load_bzip, 1299
 - lib_load_lzo, 1300
 - lib_load_zlib, 1300
 - lib_version_bzip, 1300
 - lib_version_lzo, 1300
 - lib_version_zlib, 1301
- compress2_d
 - symbols.h, 2027
- COMPRESS_ENGINE_BZIP
 - compress.h, 1294
- COMPRESS_ENGINE_LZO
 - compress.h, 1294
- COMPRESS_ENGINE_ZLIB
 - compress.h, 1294
- compress_alloc
 - compress.c, 1288
 - compress.h, 1294
- compress_block_length
 - compress.c, 1289
 - compress.h, 1294
- compress_body_data
 - compress.c, 1289
 - compress.h, 1294
- compress_body_hash
 - compress.c, 1289
 - compress.h, 1295
- compress_body_length
 - compress.c, 1289
 - compress.h, 1295
- compress_body_offset
 - compress.c, 1290
 - compress.h, 1295
- compress_bzip
 - bzip.c, 1286
 - compress.h, 1295
- compress_cleanup
 - compress.c, 1290
 - compress.h, 1296
- COMPRESS_ENGINE
 - compress.h, 1301
- compress_free
 - compress.c, 1290

- compress.h, [1296](#)
- compress_head_t
 - compress.h, [1301](#)
- compress_import
 - compress.c, [1290](#)
 - compress.h, [1296](#)
- compress_lzo
 - compress.h, [1297](#)
 - lzo.c, [1303](#)
- compress_orig_hash
 - compress.c, [1291](#)
 - compress.h, [1297](#)
- compress_orig_length
 - compress.c, [1291](#)
 - compress.h, [1297](#)
- compress_t
 - compress.h, [1293](#)
- compress_total_length
 - compress.c, [1291](#)
 - compress.h, [1297](#)
- compress_zlib
 - compress.h, [1298](#)
 - zlib.c, [1305](#)
- compressBound_d
 - symbols.h, [2027](#)
- compute_aes256_kek
 - crypto_pub.c, [1510](#)
- compute_sha_hash
 - misc_pub.c, [1557](#)
- compute_sha_hash_multibuf
 - misc_pub.c, [1557](#)
- con
 - dmtplib_session_t, [55](#)
- con_addr
 - addresses.c, [825](#)
 - network/network.h, [937](#)
- con_addr_octet
 - addresses.c, [825](#)
 - network/network.h, [937](#)
- con_addr_presentation
 - addresses.c, [826](#)
 - network/network.h, [937](#)
- con_addr_reversed
 - addresses.c, [826](#)
 - network/network.h, [938](#)
- con_addr_segment
 - addresses.c, [826](#)
 - network/network.h, [938](#)
- con_addr_standard
 - addresses.c, [827](#)
 - network/network.h, [938](#)
- con_addr_subnet
 - addresses.c, [827](#)
 - network/network.h, [939](#)
- con_addr_word
 - addresses.c, [827](#)
 - network/network.h, [939](#)

- con_decrement_refs
 - connections.c, [831](#)
 - network/network.h, [939](#)
- con_destroy
 - connections.c, [831](#)
 - network/network.h, [939](#)
- con_flush
 - connections.c, [832](#)
 - network/network.h, [940](#)
- con_increment_refs
 - connections.c, [832](#)
 - network/network.h, [940](#)
- con_init
 - connections.c, [832](#)
 - network/network.h, [940](#)
- con_init_network_buffer
 - connections.c, [833](#)
 - network/network.h, [941](#)
- con_localhost
 - connections.c, [833](#)
 - network/network.h, [941](#)
- con_print
 - network/network.h, [941](#)
 - write.c, [1010](#)
- con_private
 - connections.c, [833](#)
 - network/network.h, [942](#)
- con_read
 - network/network.h, [942](#)
 - read.c, [967](#)
- con_read_line
 - network/network.h, [942](#)
 - read.c, [968](#)
- con_reverse_check
 - network/network.h, [943](#)
 - reverse.c, [969](#)
- con_reverse_domain
 - network/network.h, [943](#)
 - reverse.c, [969](#)
- con_reverse_enqueue
 - network/network.h, [944](#)
 - reverse.c, [970](#)
- con_reverse_lookup
 - network/network.h, [944](#)
 - reverse.c, [970](#)
- con_reverse_status
 - network/network.h, [944](#)
 - reverse.c, [970](#)
- con_secure
 - connections.c, [833](#)
 - network/network.h, [944](#)
- con_status
 - connections.c, [834](#)
 - network/network.h, [945](#)
- con_write_bl
 - network/network.h, [945](#)
 - write.c, [1010](#)

- con_write_ns
 - network/network.h, 945
 - write.c, 1011
- con_write_pl
 - network/network.h, 946
 - write.c, 1011
- con_write_st
 - network/network.h, 946
 - write.c, 1012
- CONF_modules_unload_d
 - symbols.c, 1987
 - symbols.h, 2027
- config
 - magma_t, 92
- CONFIG_CHECK_DIR_READABLE
 - engine/config/config.h, 672
- CONFIG_CHECK_DIR_READWRITE
 - engine/config/config.h, 672
- CONFIG_CHECK_EXISTS
 - engine/config/config.h, 673
- CONFIG_CHECK_FILE_READABLE
 - engine/config/config.h, 673
- CONFIG_CHECK_FILE_READWRITE
 - engine/config/config.h, 673
- CONFIG_CHECK_READWRITE
 - engine/config/config.h, 673
- CONFIG_DEFAULT
 - symbols.h, 2023
- config_fetch_host_number
 - engine/config/global/datatier.c, 682
 - global.h, 730
- config_fetch_settings
 - engine/config/global/datatier.c, 682
 - global.h, 730
- config_free
 - global.c, 723
 - global.h, 730
- config_key_lookup
 - global.c, 723
 - global.h, 730
- config_load_cmdline_settings
 - global.c, 723
 - global.h, 731
- config_load_database_settings
 - global.c, 724
 - global.h, 731
- config_load_defaults
 - global.c, 724
 - global.h, 732
- config_load_file_settings
 - global.c, 724
 - global.h, 732
- config_output_help
 - global.c, 725
 - global.h, 732
- config_output_settings
 - global.c, 725
- global.h, 732
- config_output_value
 - global.c, 725
 - global.h, 733
- config_output_value_generic
 - global.c, 725
 - global.h, 733
- config_validate_settings
 - global.c, 726
 - global.h, 733
- config_value_set
 - global.c, 726
 - global.h, 734
- connect_host
 - network/pub.c, 1563
- CONNECT_TIMEOUT
 - network.c, 1561
- connection
 - __attribute__, 18
- connection_t
 - network/network.h, 950
- connections
 - magma_t, 92
- connections.c
 - con_decrement_refs, 831
 - con_destroy, 831
 - con_flush, 832
 - con_increment_refs, 832
 - con_init, 832
 - con_init_network_buffer, 833
 - con_localhost, 833
 - con_private, 833
 - con_secure, 833
 - con_status, 834
- CONSTANT
 - strings.h, 539
- CONSTANT_T
 - strings.h, 543
- constant_t
 - strings.h, 577
- consumers.h
 - __attribute__, 1309
 - cache_add, 1309
 - cache_append, 1309
 - cache_decrement, 1309
 - cache_delete, 1310
 - cache_flush, 1310
 - cache_get, 1310
 - cache_get_u64, 1310
 - cache_increment, 1311
 - cache_set, 1311
 - cache_set_u64, 1312
 - cache_silent_add, 1312
 - cache_start, 1312
 - cache_stop, 1312
 - deserialize_count_digits, 1313
 - deserialize_int16, 1313

- deserialize_int32, [1313](#)
- deserialize_int64, [1313](#)
- deserialize_ns, [1313](#)
- deserialize_ssz, [1314](#)
- deserialize_st, [1314](#)
- deserialize_sz, [1315](#)
- deserialize_uint16, [1315](#)
- deserialize_uint32, [1315](#)
- deserialize_uint64, [1315](#)
- lib_load_cache, [1316](#)
- lib_version_cache, [1316](#)
- serialization_t, [1320](#)
- serialize_int16, [1316](#)
- serialize_int32, [1316](#)
- serialize_int64, [1317](#)
- serialize_ns, [1317](#)
- serialize_ssz, [1318](#)
- serialize_st, [1318](#)
- serialize_sz, [1318](#)
- serialize_uint16, [1319](#)
- serialize_uint32, [1319](#)
- serialize_uint64, [1320](#)
- contact
 - magma_t, [92](#)
- contact.c
 - contact_print_form, [2166](#)
 - contact_print_message, [2166](#)
 - contact_process, [2167](#)
- contact.h
 - contact_abuse_checks, [2168](#)
 - contact_abuse_increment_history, [2168](#)
 - contact_business, [2169](#)
 - contact_business_add_error, [2169](#)
 - contact_business_valid_email, [2169](#)
 - contact_print_form, [2170](#)
 - contact_print_message, [2170](#)
 - contact_process, [2171](#)
- contact/abuse.c
 - contact_abuse_checks, [2158](#)
 - contact_abuse_increment_history, [2158](#)
- contact/business.c
 - contact_business, [2162](#)
 - contact_business_add_error, [2162](#)
 - contact_business_valid_email, [2163](#)
- CONTACT_DETAIL_FLAG_CRITICAL
 - contacts.h, [1045](#)
- CONTACT_DETAIL_FLAG_NONE
 - contacts.h, [1045](#)
- contact_abuse_checks
 - contact.h, [2168](#)
 - contact/abuse.c, [2158](#)
- contact_abuse_increment_history
 - contact.h, [2168](#)
 - contact/abuse.c, [2158](#)
- contact_alloc
 - contacts.h, [1045](#)
 - objects/contacts/contacts.c, [1034](#)
- contact_business
 - contact.h, [2169](#)
 - contact/business.c, [2162](#)
- contact_business_add_error
 - contact.h, [2169](#)
 - contact/business.c, [2162](#)
- contact_business_valid_email
 - contact.h, [2169](#)
 - contact/business.c, [2163](#)
- contact_create
 - contacts.h, [1046](#)
 - objects/contacts/contacts.c, [1035](#)
- contact_delete
 - contacts.h, [1046](#)
 - objects/contacts/datatier.c, [687](#)
- contact_detail_alloc
 - contacts.h, [1046](#)
 - objects/contacts/contacts.c, [1035](#)
- contact_detail_delete
 - contacts.h, [1047](#)
 - objects/contacts/datatier.c, [687](#)
- contact_detail_free
 - contacts.h, [1047](#)
 - objects/contacts/contacts.c, [1035](#)
- contact_detail_t
 - contacts.h, [1053](#)
- contact_detail_upsert
 - contacts.h, [1047](#)
 - objects/contacts/datatier.c, [688](#)
- contact_details_fetch
 - contacts.h, [1048](#)
 - objects/contacts/datatier.c, [688](#)
- contact_edit
 - contacts.h, [1048](#)
 - objects/contacts/contacts.c, [1036](#)
- contact_find_detail
 - contacts.h, [1049](#)
 - contacts/find.c, [1055](#)
- contact_find_name
 - contacts.h, [1049](#)
 - contacts/find.c, [1055](#)
- contact_find_number
 - contacts.h, [1049](#)
 - contacts/find.c, [1055](#)
- contact_folder_alloc
 - folders.h, [1075](#)
 - objects/folders/contacts.c, [1040](#)
- contact_folder_create
 - folders.h, [1075](#)
 - objects/folders/contacts.c, [1040](#)
- contact_folder_free
 - folders.h, [1076](#)
 - objects/folders/contacts.c, [1041](#)
- contact_folder_remove
 - folders.h, [1076](#)
 - objects/folders/contacts.c, [1041](#)
- contact_folder_rename

- folders.h, 1076
- objects/folders/contacts.c, 1042
- contact_folder_t
 - folders.h, 1074
- contact_free
 - contacts.h, 1049
 - objects/contacts/contacts.c, 1036
- contact_insert
 - contacts.h, 1050
 - objects/contacts/datatier.c, 688
- contact_move
 - contacts.h, 1050
 - objects/contacts/contacts.c, 1036
- contact_name
 - contacts.h, 1050
 - objects/contacts/contacts.c, 1037
- contact_print_form
 - contact.c, 2166
 - contact.h, 2170
- contact_print_message
 - contact.c, 2166
 - contact.h, 2170
- contact_process
 - contact.c, 2167
 - contact.h, 2171
- contact_remove
 - contacts.h, 1051
 - objects/contacts/contacts.c, 1037
- contact_t
 - contacts.h, 1053
- contact_update
 - contacts.h, 1051
 - objects/contacts/datatier.c, 689
- contact_update_stamp
 - contacts.h, 1052
 - objects/contacts/datatier.c, 689
- contact_validate_detail
 - contacts.h, 1052
 - objects/contacts/contacts.c, 1038
- contact_validate_name
 - contacts.h, 1052
 - objects/contacts/contacts.c, 1038
- contacts.h
 - __attribute__, 1045
 - CONTACT_DETAIL_FLAG_CRITICAL, 1045
 - CONTACT_DETAIL_FLAG_NONE, 1045
 - contact_alloc, 1045
 - contact_create, 1046
 - contact_delete, 1046
 - contact_detail_alloc, 1046
 - contact_detail_delete, 1047
 - contact_detail_free, 1047
 - contact_detail_t, 1053
 - contact_detail_upsert, 1047
 - contact_details_fetch, 1048
 - contact_edit, 1048
 - contact_find_detail, 1049
 - contact_find_name, 1049
 - contact_find_number, 1049
 - contact_free, 1049
 - contact_insert, 1050
 - contact_move, 1050
 - contact_name, 1050
 - contact_remove, 1051
 - contact_t, 1053
 - contact_update, 1051
 - contact_update_stamp, 1052
 - contact_validate_detail, 1052
 - contact_validate_name, 1052
 - contacts_fetch, 1053
 - contacts_update, 1053
- contacts/find.c
 - contact_find_detail, 1055
 - contact_find_name, 1055
 - contact_find_number, 1055
- contacts_fetch
 - contacts.h, 1053
 - objects/contacts/datatier.c, 689
- contacts_update
 - contacts.h, 1053
 - objects/contacts/contacts.c, 1038
- content
 - content.c, 2109
 - http_page_t, 77
 - magma_t, 92
- content.c
 - content, 2109
 - fonts, 2109
 - http_content_load_directory, 2106
 - http_content_load_fonts, 2106
 - http_content_refresh, 2106
 - http_content_start, 2106
 - http_content_stop, 2107
 - http_free_content, 2107
 - http_get_static, 2107
 - http_get_template, 2108
 - http_load_file, 2108
 - http_page_free, 2108
 - http_page_get, 2109
 - pages, 2109
 - templates, 2109
- context
 - server_t, 130
- context.h
 - args_parse, 763
 - display_usage, 763
 - process_start, 763
 - process_stop, 764
 - sanity_check, 764
 - signal_segfault, 764
 - signal_shutdown, 765
 - signal_start, 765
 - signal_status, 766
 - signal_thread_start, 766

- system_change_root_directory, 766
- system_fork_daemon, 766
- system_init_core_dumps, 767
- system_init_impersonation, 767
- system_init_resource_limits, 767
- system_init_umask, 768
- system_ulimit_cur, 768
- system_ulimit_max, 768
- thread_start, 769
- thread_stop, 769
- CONTIGUOUS
 - strings.h, 543
- controller.h
 - dequeue, 777
 - enqueue, 777
 - protocol_init, 778
 - protocol_process, 778
 - queue_init, 778
 - queue_shutdown, 779
 - queue_signal, 779
 - requeue, 779
- cookie
 - __attribute__, 18
- core.h
 - __attribute__, 230
 - __bool_t_defined, 225
 - __int24_t, 231
 - __uint24_t, 231
 - bool_t, 229
 - byte_t, 229
 - chr_t, 229
 - EMPTY, 229
 - INT24_MAX, 225
 - INT24_MIN, 225
 - int24_t, 229
 - int_t, 229
 - log_check, 225
 - log_critical, 225
 - log_enabled, 231
 - log_error, 225
 - log_info, 226
 - log_mutex, 231
 - log_options, 226
 - log_pedantic, 226
 - M_TYPE_BLOCK, 230
 - M_TYPE_BOOLEAN, 230
 - M_TYPE_DOUBLE, 230
 - M_TYPE_EMPTY, 230
 - M_TYPE_ENUM, 230
 - M_TYPE_FLOAT, 230
 - M_TYPE_INT16, 230
 - M_TYPE_INT32, 230
 - M_TYPE_INT64, 230
 - M_TYPE_INT8, 230
 - M_TYPE_MULTI, 230
 - M_TYPE_NULLER, 230
 - M_TYPE_PLACER, 230
 - M_TYPE_STRINGER, 230
 - M_TYPE_UINT16, 230
 - M_TYPE_UINT32, 230
 - M_TYPE_UINT64, 230
 - M_TYPE_UINT8, 230
 - M_TYPE, 229
 - SIGUNUSED, 228
 - type, 230
 - uchr_t, 229
 - UINT24_MAX, 228
 - UINT24_MIN, 228
 - uint24_t, 229
 - uint_t, 229
- core/compare/search.c
 - st_search_chr, 217
 - st_search_ci, 217
 - st_search_cs, 217
- core/host/errors.c
 - errno_name, 269
 - MAGMA_E2BIG, 266
 - MAGMA_EACCES, 266
 - MAGMA_EAGAIN, 266
 - MAGMA_EBADF, 266
 - MAGMA_EBUSY, 266
 - MAGMA_ECHILD, 266
 - MAGMA_EDOM, 266
 - MAGMA_EEXIST, 266
 - MAGMA_EFAULT, 266
 - MAGMA_EFBIG, 266
 - MAGMA_EINTR, 267
 - MAGMA_EINVAL, 267
 - MAGMA_EIO, 267
 - MAGMA_EISDIR, 267
 - MAGMA_EMFILE, 267
 - MAGMA_EMLINK, 267
 - MAGMA_ENFILE, 267
 - MAGMA_ENODEV, 267
 - MAGMA_ENOENT, 267
 - MAGMA_ENOEXEC, 268
 - MAGMA_ENOMEM, 268
 - MAGMA_ENOSPC, 268
 - MAGMA_ENOTBLK, 268
 - MAGMA_ENOTDIR, 268
 - MAGMA_ENOTTY, 268
 - MAGMA_ENXIO, 268
 - MAGMA_EPERM, 268
 - MAGMA_EPIPE, 268
 - MAGMA_ERANGE, 269
 - MAGMA_EROFS, 269
 - MAGMA_ESPIPE, 269
 - MAGMA_ESRCH, 269
 - MAGMA_ETXTBSY, 269
 - MAGMA_EXDEV, 269
- core/host/process.c
 - process_find_pid, 314
 - process_kill, 314
 - process_my_pid, 314

- core/parsers/parsers.h
 - line_pl_bl, [444](#)
 - line_pl_ns, [444](#)
 - line_pl_pl, [444](#)
 - line_pl_st, [445](#)
 - lower_chr, [445](#)
 - lower_st, [445](#)
 - pl_get_embraced, [446](#)
 - pl_shrink_before_characters, [446](#)
 - pl_skip_characters, [446](#)
 - pl_skip_to_characters, [447](#)
 - pl_trim, [447](#)
 - pl_trim_end, [447](#)
 - pl_trim_start, [447](#)
 - pl_update_start, [448](#)
 - st_and, [448](#)
 - st_not, [448](#)
 - st_or, [449](#)
 - st_trim, [449](#)
 - st_xor, [449](#)
 - str_tok_get_bl, [450](#)
 - str_tok_get_count_bl, [450](#)
 - time_datestamp, [450](#)
 - time_print_gmt, [451](#)
 - time_print_local, [451](#)
 - time_till_midnight, [451](#)
 - tok_get_bl, [451](#)
 - tok_get_count_bl, [452](#)
 - tok_get_count_st, [452](#)
 - tok_get_ns, [453](#)
 - tok_get_pl, [453](#)
 - tok_get_st, [453](#)
 - tok_pop, [454](#)
 - tok_pop_init_bl, [454](#)
 - tok_pop_init_st, [454](#)
 - upper_chr, [454](#)
 - upper_st, [455](#)
- core/strings/data.c
 - st_char_get, [493](#)
 - st_data_get, [495](#)
 - st_data_set, [495](#)
 - st_empty_out, [496](#)
 - st_empty_variadic, [496](#)
 - st_populated_variadic, [496](#)
 - st_set, [497](#)
 - st_uchar_get, [497](#)
 - st_wipe, [497](#)
- core/thread/keys.c
 - tkey_get, [583](#)
 - tkey_init, [583](#)
 - tkey_set, [583](#)
- core/thread/thread.c
 - thread_alloc, [594](#)
 - thread_cancel, [594](#)
 - thread_cancel_disable, [595](#)
 - thread_cancel_enable, [595](#)
 - thread_get_thread_id, [595](#)
 - thread_join, [595](#)
 - thread_launch, [596](#)
 - thread_result, [596](#)
 - thread_signal, [596](#)
- core_dump_size_limit
 - magma_t, [92](#)
- count
 - engine/status/statistics.c, [804](#)
 - magma_t, [92](#), [93](#)
 - meta_stats_tag_t, [110](#)
- count_ptr_chain
 - misc_pub.c, [1557](#)
- crc.c
 - A, [194](#)
 - A1, [194](#)
 - B, [195](#)
 - C, [195](#)
 - crc24_checksum, [195](#)
 - crc24_final, [196](#)
 - CRC24_INIT, [195](#)
 - crc24_init, [196](#)
 - CRC24_POLY, [195](#)
 - crc24_table, [198](#)
 - crc24_update, [196](#)
 - crc32_checksum, [196](#)
 - crc32_table, [198](#)
 - crc32_update, [196](#)
 - crc64_checksum, [197](#)
 - crc64_table, [198](#)
 - crc64_update, [197](#)
 - D, [195](#)
 - S32, [195](#)
 - S8, [195](#)
- crc24_checksum
 - checksum.h, [190](#)
 - crc.c, [195](#)
- crc24_final
 - checksum.h, [191](#)
 - crc.c, [196](#)
- CRC24_INIT
 - crc.c, [195](#)
- crc24_init
 - checksum.h, [191](#)
 - crc.c, [196](#)
- CRC24_POLY
 - crc.c, [195](#)
- crc24_table
 - crc.c, [198](#)
- crc24_update
 - checksum.h, [191](#)
 - crc.c, [196](#)
- crc32_checksum
 - checksum.h, [191](#)
 - crc.c, [196](#)
- crc32_table
 - crc.c, [198](#)
- crc32_update

- checksum.h, 191
- crc.c, 196
- crc64_checksum
 - checksum.h, 192
 - crc.c, 197
- crc64_table
 - crc.c, 198
- crc64_update
 - checksum.h, 192
 - crc.c, 197
- creation.c
 - stacie_create_nonce, 1940
 - stacie_create_salt, 1940
 - stacie_create_shard, 1941
- CREDENTIAL_AUTH
 - network/meta.h, 905
- CREDENTIAL_MAIL
 - network/meta.h, 905
- CRL_FILE
 - signet-ssl.h, 1770
- cryptex_alloc
 - deprecated.h, 1496
 - deprecated/cryptex.c, 1338
- cryptex_body_data
 - deprecated.h, 1496
 - deprecated/cryptex.c, 1338
- cryptex_body_length
 - deprecated.h, 1496
 - deprecated/cryptex.c, 1339
- cryptex_envelope_data
 - deprecated.h, 1497
 - deprecated/cryptex.c, 1339
- cryptex_envelope_length
 - deprecated.h, 1497
 - deprecated/cryptex.c, 1339
- cryptex_free
 - deprecated.h, 1497
 - deprecated/cryptex.c, 1340
- cryptex_head_t
 - cryptography/cryptography.h, 1379
- cryptex_hmac_data
 - deprecated.h, 1498
 - deprecated/cryptex.c, 1340
- cryptex_hmac_length
 - deprecated.h, 1498
 - deprecated/cryptex.c, 1340
- cryptex_original_length
 - deprecated.h, 1498
 - deprecated/cryptex.c, 1340
- cryptex_t
 - cryptography/cryptography.h, 1349
- cryptex_total_length
 - deprecated.h, 1498
 - deprecated/cryptex.c, 1341
- crypto.h
 - __attribute__, 1583
 - dime_dmsg_actor_to_string, 1583
 - dime_dmsg_chunks_sig_origin_sign, 1583
 - dime_dmsg_kek_in_derive, 1584
 - dime_dmsg_message_binary_deserialize, 1584
 - dime_dmsg_message_binary_serialize, 1584
 - dime_dmsg_message_decrypt_as_auth, 1585
 - dime_dmsg_message_decrypt_as_dest, 1585
 - dime_dmsg_message_decrypt_as_orig, 1585
 - dime_dmsg_message_decrypt_as_recp, 1586
 - dime_dmsg_message_destroy, 1586
 - dime_dmsg_message_encrypt, 1586
 - dime_dmsg_message_envelope_decrypt, 1586
 - dime_dmsg_message_state_get, 1587
 - dime_dmsg_object_chunk_create, 1587
 - dime_dmsg_object_chunklist_destroy, 1587
 - dime_dmsg_object_destroy, 1587
 - dime_dmsg_object_dump, 1588
 - dime_dmsg_object_state_init, 1588
 - dime_dmsg_object_state_to_string, 1588
 - DMIME_OBJECT_STATE_COMPLETE, 1583
 - DMIME_OBJECT_STATE_CREATION, 1583
 - DMIME_OBJECT_STATE_INCOMPLETE_ENVELOPE, 1583
 - DMIME_OBJECT_STATE_INCOMPLETE_METADATA, 1583
 - DMIME_OBJECT_STATE_LOADED_ENVELOPE, 1583
 - DMIME_OBJECT_STATE_LOADED_SIGNETS, 1583
 - DMIME_OBJECT_STATE_NONE, 1583
 - dmime_actor_t, 1582
 - dmime_kek_t, 1589
 - dmime_message_chunk_state_t, 1582
 - dmime_message_chunk_t, 1589
 - dmime_message_state_t, 1582
 - dmime_object_chunk_t, 1582
 - dmime_object_state_t, 1583
 - dmime_tracing_t, 1589
 - id_author, 1582
 - id_destination, 1582
 - id_origin, 1582
 - id_recipient, 1582
 - MESSAGE_CHUNK_STATE_CREATION, 1582
 - MESSAGE_CHUNK_STATE_ENCODED, 1582
 - MESSAGE_CHUNK_STATE_ENCRYPTED, 1582
 - MESSAGE_CHUNK_STATE_NONE, 1582
 - MESSAGE_CHUNK_STATE_SIGNED, 1582
 - MESSAGE_CHUNK_STATE_UNKNOWN, 1582
 - MESSAGE_STATE_AUTHOR_SIGNED, 1583
 - MESSAGE_STATE_CHUNKS_SIGNED, 1582
 - MESSAGE_STATE_COMPLETE, 1583
 - MESSAGE_STATE_EMPTY, 1582
 - MESSAGE_STATE_ENCODED, 1582
 - MESSAGE_STATE_ENCRYPTED, 1582
 - MESSAGE_STATE_INCOMPLETE, 1582
 - MESSAGE_STATE_NONE, 1582
 - TRACING_LENGTH_SIZE, 1582
- CRYPTO_cleanup_all_ex_data_d
 - symbols.c, 1988
 - symbols.h, 2027

- CRYPTO_free_d
 - symbols.c, [1988](#)
 - symbols.h, [2027](#)
- crypto_init
 - crypto_pub.c, [1510](#)
- crypto_int32
 - curve25519-ref10.c, [1671](#)
 - ed25519-ref10.c, [1677](#)
- crypto_int64
 - curve25519-ref10.c, [1671](#)
 - ed25519-ref10.c, [1677](#)
- CRYPTO_num_locks_d
 - symbols.c, [1988](#)
 - symbols.h, [2027](#)
- crypto_pub.c
 - compute_aes256_kek, [1510](#)
 - crypto_init, [1510](#)
 - crypto_shutdown, [1510](#)
 - decrypt_aes_256, [1511](#)
 - deserialize_ec_privkey, [1511](#)
 - deserialize_ec_pubkey, [1511](#)
 - deserialize_ed25519_privkey, [1511](#)
 - deserialize_ed25519_pubkey, [1511](#)
 - ec_sign_data, [1511](#)
 - ec_sign_sha_data, [1511](#)
 - ecies_env_derivation, [1511](#)
 - ed25519_sign_data, [1511](#)
 - ed25519_verify_sig, [1512](#)
 - encrypt_aes_256, [1512](#)
 - free_ec_key, [1512](#)
 - free_ed25519_key, [1512](#)
 - free_ed25519_key_chain, [1512](#)
 - generate_ec_keypair, [1512](#)
 - generate_ed25519_keypair, [1512](#)
 - get_random_bytes, [1512](#)
 - load_ec_privkey, [1512](#)
 - load_ec_pubkey, [1513](#)
 - load_ed25519_privkey, [1513](#)
 - serialize_ec_privkey, [1513](#)
 - serialize_ec_pubkey, [1513](#)
 - verify_ec_sha_signature, [1513](#)
 - verify_ec_signature, [1513](#)
- crypto_scalarmult_base_ref10
 - curve25519-ref10.c, [1672](#)
 - curve25519-ref10.h, [1674](#)
- crypto_scalarmult_ref10
 - curve25519-ref10.c, [1672](#)
 - curve25519-ref10.h, [1674](#)
- CRYPTO_set_id_callback_d
 - symbols.h, [2027](#)
- CRYPTO_set_locked_mem_functions_d
 - symbols.h, [2027](#)
- CRYPTO_set_locking_callback_d
 - symbols.h, [2027](#)
- CRYPTO_set_mem_functions_d
 - symbols.h, [2028](#)
- crypto_shutdown
 - crypto_pub.c, [1510](#)
- crypto_sign_open_ref10
 - ed25519-ref10.c, [1678](#)
 - ed25519-ref10.h, [1679](#)
- crypto_sign_pk_ref10
 - ed25519-ref10.c, [1678](#)
 - ed25519-ref10.h, [1679](#)
- crypto_sign_ref10
 - ed25519-ref10.c, [1678](#)
 - ed25519-ref10.h, [1679](#)
- crypto_uint32
 - ed25519-ref10.c, [1677](#)
- crypto_uint64
 - curve25519-ref10.c, [1671](#)
 - ed25519-ref10.c, [1678](#)
- cryptography
 - magma_t, [93](#)
- cryptography/cryptex.c
 - deprecated_cryptex_alloc, [1334](#)
 - deprecated_cryptex_body_data, [1334](#)
 - deprecated_cryptex_body_length, [1335](#)
 - deprecated_cryptex_envelope_data, [1335](#)
 - deprecated_cryptex_envelope_length, [1335](#)
 - deprecated_cryptex_free, [1336](#)
 - deprecated_cryptex_hmac_data, [1336](#)
 - deprecated_cryptex_hmac_length, [1336](#)
 - deprecated_cryptex_original_length, [1336](#)
 - deprecated_cryptex_total_length, [1337](#)
- cryptography/cryptography.h
 - __attribute__, [1350](#)
 - BN_num_bytes_d, [1348](#)
 - cipher_block_length, [1350](#)
 - cipher_id, [1350](#)
 - cipher_key_length, [1350](#)
 - cipher_name, [1351](#)
 - cipher_numeric_id, [1351](#)
 - cipher_t, [1349](#)
 - cipher_vector_length, [1351](#)
 - cryptex_head_t, [1379](#)
 - cryptex_t, [1349](#)
 - deprecated_cryptex_alloc, [1351](#)
 - deprecated_cryptex_body_data, [1352](#)
 - deprecated_cryptex_body_length, [1352](#)
 - deprecated_cryptex_envelope_data, [1352](#)
 - deprecated_cryptex_envelope_length, [1352](#)
 - deprecated_cryptex_free, [1353](#)
 - deprecated_cryptex_hmac_data, [1353](#)
 - deprecated_cryptex_hmac_length, [1353](#)
 - deprecated_cryptex_original_length, [1354](#)
 - deprecated_cryptex_total_length, [1354](#)
 - deprecated_ecies_decrypt, [1354](#)
 - deprecated_ecies_encrypt, [1355](#)
 - deprecated_ecies_envelope_derivation, [1355](#)
 - deprecated_ecies_group, [1355](#)
 - deprecated_ecies_key_alloc, [1355](#)
 - deprecated_ecies_key_create, [1356](#)
 - deprecated_ecies_key_free, [1356](#)

deprecated_ecies_key_private, 1356
deprecated_ecies_key_private_bin, 1356
deprecated_ecies_key_private_hex, 1357
deprecated_ecies_key_public, 1357
deprecated_ecies_key_public_bin, 1357
deprecated_ecies_key_public_hex, 1357
deprecated_ecies_start, 1358
deprecated_ecies_stop, 1358
deprecated_hmac_digest, 1358
deprecated_hmac_md4, 1358
deprecated_hmac_md5, 1359
deprecated_hmac_ripemd160, 1359
deprecated_hmac_sha, 1359
deprecated_hmac_sha1, 1359
deprecated_hmac_sha224, 1359
deprecated_hmac_sha256, 1359
deprecated_hmac_sha384, 1359
deprecated_hmac_sha512, 1359
deprecated_scramble_alloc, 1359
deprecated_scramble_body_data, 1360
deprecated_scramble_body_hash, 1360
deprecated_scramble_body_length, 1360
deprecated_scramble_cleanup, 1361
deprecated_scramble_decrypt, 1361
deprecated_scramble_encrypt, 1361
deprecated_scramble_free, 1362
deprecated_scramble_import, 1362
deprecated_scramble_orig_length, 1362
deprecated_scramble_total_length, 1363
deprecated_scramble_vector_data, 1363
deprecated_scramble_vector_length, 1363
deprecated_symmetric_decrypt, 1363
deprecated_symmetric_encrypt, 1364
deprecated_symmetric_key, 1364
deprecated_symmetric_vector, 1364
dh_exchange_2048, 1365
dh_exchange_4096, 1365
dh_params_2048, 1366
dh_params_4096, 1366
dh_params_generate, 1366
dh_params_generate_callback, 1366
dh_static_2048, 1366
dh_static_4096, 1366
digest_id, 1366
digest_name, 1367
digest_t, 1349
ECIES_PRIVATE_BINARY, 1349
ECIES_PRIVATE_HEX, 1349
ECIES_PUBLIC_BINARY, 1349
ECIES_PUBLIC_HEX, 1349
ECIES_CIPHER, 1348
ECIES_CURVE, 1348
ECIES_ENVELOPE, 1348
ECIES_HMAC, 1348
ECIES_KEY_TYPE, 1349
hash_digest, 1367
hash_md4, 1367
hash_md5, 1367
hash_ripemd160, 1367
hash_sha, 1367
hash_sha1, 1367
hash_sha224, 1367
hash_sha256, 1368
hash_sha384, 1368
hash_sha512, 1368
lib_load_openssl, 1368
lib_version_openssl, 1368
MAGMA_CIPHERS_GENERIC, 1348
MAGMA_CIPHERS_HIGH, 1348
MAGMA_CIPHERS_LOW, 1348
MAGMA_CIPHERS_MEDIUM, 1349
OPENSSL_free_d, 1349
rand_choices, 1368
rand_get_int16, 1369
rand_get_int32, 1369
rand_get_int64, 1369
rand_get_int8, 1369
rand_get_uint16, 1369
rand_get_uint32, 1370
rand_get_uint64, 1370
rand_get_uint8, 1370
rand_start, 1371
rand_stop, 1371
rand_thread_start, 1371
rand_write, 1371
scramble_head_t, 1379
scramble_t, 1349
ssl_dh2048_exchange_callback, 1372
ssl_dh4096_exchange_callback, 1372
ssl_dh_generate_callback, 1372
ssl_ecdh_exchange_callback, 1372
ssl_error_string, 1372
ssl_locking_callback, 1373
ssl_start, 1373
ssl_stop, 1373
ssl_thread_id_callback, 1374
ssl_thread_stop, 1374
ssl_verify_privkey, 1374
TLS, 1349
tls_bits, 1375
tls_cipher, 1375
tls_client_alloc, 1375
tls_continue, 1376
tls_error, 1376
tls_free, 1376
tls_print, 1376
tls_read, 1377
tls_server_alloc, 1377
tls_server_create, 1377
tls_server_destroy, 1378
tls_status, 1378
tls_suite, 1378
tls_version, 1379
tls_write, 1379

cryptography/ecies.c

- deprecated_ecies_decrypt, 1391
- deprecated_ecies_encrypt, 1391
- deprecated_ecies_envelope_derivation, 1391
- deprecated_ecies_group, 1392
- deprecated_ecies_key_alloc, 1392
- deprecated_ecies_key_create, 1392
- deprecated_ecies_key_free, 1392
- deprecated_ecies_key_private, 1392
- deprecated_ecies_key_private_bin, 1393
- deprecated_ecies_key_private_hex, 1393
- deprecated_ecies_key_public, 1393
- deprecated_ecies_key_public_bin, 1393
- deprecated_ecies_key_public_hex, 1394
- deprecated_ecies_start, 1394
- deprecated_ecies_stop, 1394
- ecies_cipher_evp, 1394
- ecies_curve_group, 1394
- ecies_envelope_evp, 1395
- ecies_hmac_evp, 1395

cryptography/hmac.c

- deprecated_hmac_digest, 1404
- deprecated_hmac_md4, 1404
- deprecated_hmac_md5, 1404
- deprecated_hmac_ripemd160, 1404
- deprecated_hmac_sha, 1405
- deprecated_hmac_sha1, 1405
- deprecated_hmac_sha224, 1405
- deprecated_hmac_sha256, 1405
- deprecated_hmac_sha384, 1405
- deprecated_hmac_sha512, 1405

cryptography/scramble.c

- deprecated_scramble_alloc, 1422
- deprecated_scramble_body_data, 1423
- deprecated_scramble_body_hash, 1423
- deprecated_scramble_body_length, 1423
- deprecated_scramble_cleanup, 1423
- deprecated_scramble_decrypt, 1424
- deprecated_scramble_encrypt, 1424
- deprecated_scramble_free, 1424
- deprecated_scramble_import, 1425
- deprecated_scramble_orig_length, 1425
- deprecated_scramble_total_length, 1425
- deprecated_scramble_vector_data, 1426
- deprecated_scramble_vector_length, 1426

cryptography/symmetric.c

- deprecated_symmetric_decrypt, 1432
- deprecated_symmetric_encrypt, 1432
- deprecated_symmetric_key, 1432
- deprecated_symmetric_vector, 1433

cursors.c

- inx_cursor_alloc, 326
- inx_cursor_free, 327
- inx_cursor_key_active, 327
- inx_cursor_key_next, 327
- inx_cursor_reset, 328
- inx_cursor_value_active, 328

- inx_cursor_value_next, 328

curve25519-donna-32bit.h

- bignum25519, 1638
- bignum25519align16, 1638
- carry_pass, 1637
- carry_pass_final, 1637
- carry_pass_full, 1637
- curve25519_mul_noinline, 1637
- F, 1637, 1638

curve25519-donna-64bit.h

- bignum25519, 1640
- curve25519_contract_carry, 1639
- curve25519_contract_carry_final, 1639
- curve25519_contract_carry_full, 1639
- ED25519_64BIT_TABLES, 1639
- F, 1639
- write51, 1640
- write51full, 1640

curve25519-donna-sse2.h

- bignum25519, 1643
- carry_pass, 1642
- carry_pass_final, 1642
- carry_pass_full, 1642
- curve25519_add_after_basic, 1643
- curve25519_square, 1643
- F, 1643
- packed32bignum25519, 1643
- packed64bignum25519, 1643
- packedelem32, 1643
- packedelem64, 1643
- packedelem8, 1643
- xmmi, 1643

curve25519-ref10.c

- crypto_int32, 1671
- crypto_int64, 1671
- crypto_scalarmult_base_ref10, 1672
- crypto_scalarmult_ref10, 1672
- crypto_uint64, 1671
- fe, 1671
- fe_0, 1672
- fe_1, 1672
- fe_add, 1672
- fe_copy, 1672
- fe_cswap, 1672
- fe_frombytes, 1672
- fe_invert, 1672
- fe_mul, 1673
- fe_mul121666, 1673
- fe_sq, 1673
- fe_sub, 1673
- fe_tobytes, 1673

curve25519-ref10.h

- crypto_scalarmult_base_ref10, 1674
- crypto_scalarmult_ref10, 1674
- curve25519_add_after_basic
- curve25519-donna-sse2.h, 1643
- curve25519_contract_carry

- curve25519-donna-64bit.h, 1639
- curve25519_contract_carry_final
 - curve25519-donna-64bit.h, 1639
- curve25519_contract_carry_full
 - curve25519-donna-64bit.h, 1639
- curve25519_mul_noinline
 - curve25519-donna-32bit.h, 1637
- curve25519_square
 - curve25519-donna-sse2.h, 1643
- curved25519_expected
 - test.c, 1689
- curved25519_key
 - ed25519.h, 1670
 - fuzz/ed25519-donna.h, 1658
- curved25519_scalarmult_basepoint
 - dime/ed25519/ed25519.c, 1665
 - fuzz/ed25519-donna.h, 1659
- curved25519_scalarmult_basepoint_donna
 - ed25519.h, 1670
- curved25519_scalarmult_basepoint_sse2
 - fuzz/ed25519-donna.h, 1659
- custom_deserializer_t
 - providers/dime/signet-resolver/cache.h, 651
- custom_dumper_t
 - providers/dime/signet-resolver/cache.h, 652
- custom_serializer_t
 - providers/dime/signet-resolver/cache.h, 652
- cutoff
 - server_t, 130

D

- crc.c, 195
- d2i_ECDSA_SIG_d
 - symbols.h, 2028
- d2i_ECPrivateKey_d
 - symbols.h, 2028
- d2i_OCSP_RESPONSE_d
 - symbols.h, 2028
- daemonize
 - magma_t, 93
- daily_rcv_limit
 - smtp_inbound_prefs_t, 142
- daily_rcv_limit_ip
 - smtp_inbound_prefs_t, 142
- daily_send_limit
 - smtp_outbound_prefs_t, 149
- dash
 - mysql.c, 1470
- data
 - cached_object, 31
 - object_chunk, 117
 - queue_t, 121
 - secure.c, 390
 - sha_datbuf_t, 133
 - signet_t, 138
- data_type_7bit
 - providers/dime/signet-resolver/dmtp.h, 838

- data_type_8bit
 - providers/dime/signet-resolver/dmtp.h, 838
- data_type_default
 - providers/dime/signet-resolver/dmtp.h, 838
- data_offset
 - signet_field_t, 136
- DATA_SEGMENT_CONTINUATION_ENABLED
 - dmessage/common.h, 1570
- data_size
 - object_chunk, 117
 - signet_field_key_t, 134
 - signet_field_t, 136
- data_true
 - secure.c, 390
- data_type
 - signet_field_key_t, 134
- database
 - magma_keys_t, 86
 - magma_t, 93
- database.h
 - ISNULL, 1445
 - lib_load_mysql, 1446
 - lib_version_mysql, 1446
 - res_bind_create, 1446
 - res_bind_free, 1446
 - res_field_block, 1446
 - res_field_bool, 1447
 - res_field_count, 1447
 - res_field_double, 1447
 - res_field_float, 1448
 - res_field_generic, 1448
 - res_field_int16, 1448
 - res_field_int32, 1448
 - res_field_int64, 1449
 - res_field_int8, 1449
 - res_field_length, 1449
 - res_field_string, 1450
 - res_field_uint16, 1450
 - res_field_uint32, 1450
 - res_field_uint64, 1451
 - res_field_uint8, 1451
 - res_row_count, 1451
 - res_row_get, 1452
 - res_row_next, 1452
 - res_row_set, 1453
 - res_row_store, 1453
 - res_stmt_store, 1453
 - res_table_alloc, 1453
 - res_table_free, 1453
 - row_t, 1445
 - serv_charset_mysql, 1454
 - serv_schema_mysql, 1454
 - serv_type_mysql, 1454
 - serv_version_mysql, 1454
 - sql_errno, 1454
 - sql_error, 1455
 - sql_insert, 1455

- sql_insert_conn, [1455](#)
- sql_num_rows, [1455](#)
- sql_num_rows_conn, [1455](#)
- sql_open, [1455](#)
- sql_ping, [1456](#)
- sql_pool, [1465](#)
- sql_query, [1456](#)
- sql_query_conn, [1457](#)
- sql_query_res, [1457](#)
- sql_query_res_conn, [1457](#)
- sql_start, [1457](#)
- sql_stop, [1458](#)
- sql_thread_start, [1458](#)
- sql_thread_stop, [1458](#)
- sql_write, [1458](#)
- sql_write_conn, [1458](#)
- stmt_bind_param, [1459](#)
- stmt_close, [1459](#)
- stmt_errno, [1459](#)
- stmt_error, [1459](#)
- stmt_exec, [1459](#)
- stmt_exec_affected, [1460](#)
- stmt_exec_affected_conn, [1460](#)
- stmt_exec_conn, [1461](#)
- stmt_get_result, [1461](#)
- stmt_get_result_conn, [1462](#)
- stmt_insert, [1462](#)
- stmt_insert_conn, [1462](#)
- stmt_open, [1463](#)
- stmt_prepare, [1463](#)
- stmt_rebuild, [1463](#)
- stmt_reset, [1463](#)
- stmt_start, [1464](#)
- stmt_stop, [1464](#)
- table_t, [1445](#)
- tran_commit, [1464](#)
- tran_rollback, [1464](#)
- tran_start, [1465](#)
- dataset
 - test.c, [1689](#)
- date
 - mail_message_t, [104](#)
 - smtp_message_t, [147](#)
- day
 - engine/context/process.c, [317](#)
- dbgprint
 - misc_pub.c, [1557](#)
- dbl
 - multi_t, [111](#)
- dcrypto.h
 - AES_256_KEY_SIZE, [1515](#)
 - AES_256_KEY_SIZE, [1515](#)
 - AES_256_PADDING_SIZE, [1515](#)
 - EC_ENCRYPT_CURVE, [1515](#)
 - EC_PUBKEY_SIZE, [1515](#)
 - EC_SIGNING_CURVE, [1515](#)
 - ED25519_KEY_B64_SIZE, [1515](#)
 - ED25519_KEY_SIZE, [1515](#)
 - ED25519_SIG_B64_SIZE, [1516](#)
 - ED25519_SIG_SIZE, [1516](#)
 - PUBLIC_FUNC_DECL, [1518](#)
- DEBUG
 - prime.h, [1904](#)
- decode_rsa_pubkey
 - misc_pub.c, [1557](#)
- decompress_block_lzo
 - compress.h, [1298](#)
 - lzo.c, [1303](#)
- decompress_bzip
 - bzip.c, [1286](#)
 - compress.h, [1298](#)
- decompress_lzo
 - compress.h, [1299](#)
 - lzo.c, [1303](#)
- decompress_zlib
 - compress.h, [1299](#)
 - zlib.c, [1305](#)
- decrypt_aes_256
 - crypto_pub.c, [1511](#)
- DEFAULT_CHUNK_FLAGS
 - dmessage/common.h, [1570](#)
- deflate_d
 - zlib.c, [1306](#)
- deflateEnd_d
 - zlib.c, [1306](#)
- deflateInit2__d
 - zlib.c, [1306](#)
- delay
 - server_t, [130](#)
- DELETE_CONTACT
 - queries.h, [2071](#)
- DELETE_CONTACT_DETAILS
 - queries.h, [2071](#)
- DELETE_FOLDER
 - queries.h, [2071](#)
- DELETE_MESSAGE
 - queries.h, [2071](#)
- DELETE_MESSAGE_TAG
 - queries.h, [2072](#)
- DELETE_MESSAGE_TAGS
 - queries.h, [2072](#)
- DELETE_OBJECT
 - queries.h, [2072](#)
- DELETE_SIGNATURE
 - queries.h, [2072](#)
- DELETE_USER
 - queries.h, [2072](#)
- DELETE_USER_CONFIG
 - queries.h, [2072](#)
- deprecated.h
 - cryptex_alloc, [1496](#)
 - cryptex_body_data, [1496](#)
 - cryptex_body_length, [1496](#)
 - cryptex_envelope_data, [1497](#)

- cryptex_envelope_length, [1497](#)
- cryptex_free, [1497](#)
- cryptex_hmac_data, [1498](#)
- cryptex_hmac_length, [1498](#)
- cryptex_original_length, [1498](#)
- cryptex_total_length, [1498](#)
- ecies_decrypt, [1499](#)
- ecies_encrypt, [1499](#)
- ecies_envelope_derivation, [1500](#)
- ecies_group, [1500](#)
- ecies_key_alloc, [1500](#)
- ecies_key_create, [1500](#)
- ecies_key_free, [1500](#)
- ecies_key_private, [1501](#)
- ecies_key_private_bin, [1501](#)
- ecies_key_private_hex, [1501](#)
- ecies_key_public, [1501](#)
- ecies_key_public_bin, [1502](#)
- ecies_key_public_hex, [1502](#)
- ecies_start, [1502](#)
- ecies_stop, [1502](#)
- hmac_digest, [1503](#)
- hmac_md4, [1503](#)
- hmac_md5, [1503](#)
- hmac_ripemd160, [1503](#)
- hmac_sha, [1503](#)
- hmac_sha1, [1503](#)
- hmac_sha224, [1503](#)
- hmac_sha256, [1504](#)
- hmac_sha384, [1504](#)
- hmac_sha512, [1504](#)
- scramble_alloc, [1504](#)
- scramble_body_data, [1504](#)
- scramble_body_hash, [1504](#)
- scramble_body_length, [1505](#)
- scramble_cleanup, [1505](#)
- scramble_decrypt, [1505](#)
- scramble_encrypt, [1506](#)
- scramble_free, [1506](#)
- scramble_import, [1506](#)
- scramble_orig_length, [1507](#)
- scramble_total_length, [1507](#)
- scramble_vector_data, [1507](#)
- scramble_vector_length, [1507](#)
- symmetric_decrypt, [1508](#)
- symmetric_encrypt, [1508](#)
- symmetric_key, [1508](#)
- symmetric_vector, [1509](#)
- deprecated/cryptex.c
 - cryptex_alloc, [1338](#)
 - cryptex_body_data, [1338](#)
 - cryptex_body_length, [1339](#)
 - cryptex_envelope_data, [1339](#)
 - cryptex_envelope_length, [1339](#)
 - cryptex_free, [1340](#)
 - cryptex_hmac_data, [1340](#)
 - cryptex_hmac_length, [1340](#)
 - cryptex_original_length, [1340](#)
 - cryptex_total_length, [1341](#)
- deprecated/ecies.c
 - ecies_cipher_evp, [1400](#)
 - ecies_curve_group, [1400](#)
 - ecies_decrypt, [1397](#)
 - ecies_encrypt, [1397](#)
 - ecies_envelope_derivation, [1397](#)
 - ecies_envelope_evp, [1400](#)
 - ecies_group, [1397](#)
 - ecies_hmac_evp, [1400](#)
 - ecies_key_alloc, [1398](#)
 - ecies_key_create, [1398](#)
 - ecies_key_free, [1398](#)
 - ecies_key_private, [1398](#)
 - ecies_key_private_bin, [1398](#)
 - ecies_key_private_hex, [1399](#)
 - ecies_key_public, [1399](#)
 - ecies_key_public_bin, [1399](#)
 - ecies_key_public_hex, [1399](#)
 - ecies_start, [1400](#)
 - ecies_stop, [1400](#)
- deprecated/hmac.c
 - hmac_digest, [1406](#)
 - hmac_md4, [1406](#)
 - hmac_md5, [1406](#)
 - hmac_ripemd160, [1407](#)
 - hmac_sha, [1407](#)
 - hmac_sha1, [1407](#)
 - hmac_sha224, [1407](#)
 - hmac_sha256, [1407](#)
 - hmac_sha384, [1407](#)
 - hmac_sha512, [1407](#)
- deprecated/scramble.c
 - scramble_alloc, [1427](#)
 - scramble_body_data, [1428](#)
 - scramble_body_hash, [1428](#)
 - scramble_body_length, [1428](#)
 - scramble_cleanup, [1428](#)
 - scramble_decrypt, [1429](#)
 - scramble_encrypt, [1429](#)
 - scramble_free, [1429](#)
 - scramble_import, [1430](#)
 - scramble_orig_length, [1430](#)
 - scramble_total_length, [1430](#)
 - scramble_vector_data, [1431](#)
 - scramble_vector_length, [1431](#)
- deprecated/symmetric.c
 - symmetric_decrypt, [1434](#)
 - symmetric_encrypt, [1434](#)
 - symmetric_key, [1435](#)
 - symmetric_vector, [1435](#)
- deprecated_cryptex_alloc
 - cryptography/cryptex.c, [1334](#)
 - cryptography/cryptography.h, [1351](#)
- deprecated_cryptex_body_data
 - cryptography/cryptex.c, [1334](#)

[cryptography/cryptography.h](#), 1352
[deprecated_cryptex_body_length](#)
[cryptography/cryptex.c](#), 1335
[cryptography/cryptography.h](#), 1352
[deprecated_cryptex_envelope_data](#)
[cryptography/cryptex.c](#), 1335
[cryptography/cryptography.h](#), 1352
[deprecated_cryptex_envelope_length](#)
[cryptography/cryptex.c](#), 1335
[cryptography/cryptography.h](#), 1352
[deprecated_cryptex_free](#)
[cryptography/cryptex.c](#), 1336
[cryptography/cryptography.h](#), 1353
[deprecated_cryptex_hmac_data](#)
[cryptography/cryptex.c](#), 1336
[cryptography/cryptography.h](#), 1353
[deprecated_cryptex_hmac_length](#)
[cryptography/cryptex.c](#), 1336
[cryptography/cryptography.h](#), 1353
[deprecated_cryptex_original_length](#)
[cryptography/cryptex.c](#), 1336
[cryptography/cryptography.h](#), 1354
[deprecated_cryptex_total_length](#)
[cryptography/cryptex.c](#), 1337
[cryptography/cryptography.h](#), 1354
[deprecated_ecies_decrypt](#)
[cryptography/cryptography.h](#), 1354
[cryptography/ecies.c](#), 1391
[deprecated_ecies_encrypt](#)
[cryptography/cryptography.h](#), 1355
[cryptography/ecies.c](#), 1391
[deprecated_ecies_envelope_derivation](#)
[cryptography/cryptography.h](#), 1355
[cryptography/ecies.c](#), 1391
[deprecated_ecies_group](#)
[cryptography/cryptography.h](#), 1355
[cryptography/ecies.c](#), 1392
[deprecated_ecies_key_alloc](#)
[cryptography/cryptography.h](#), 1355
[cryptography/ecies.c](#), 1392
[deprecated_ecies_key_create](#)
[cryptography/cryptography.h](#), 1356
[cryptography/ecies.c](#), 1392
[deprecated_ecies_key_free](#)
[cryptography/cryptography.h](#), 1356
[cryptography/ecies.c](#), 1392
[deprecated_ecies_key_private](#)
[cryptography/cryptography.h](#), 1356
[cryptography/ecies.c](#), 1392
[deprecated_ecies_key_private_bin](#)
[cryptography/cryptography.h](#), 1356
[cryptography/ecies.c](#), 1393
[deprecated_ecies_key_private_hex](#)
[cryptography/cryptography.h](#), 1357
[cryptography/ecies.c](#), 1393
[deprecated_ecies_key_public](#)
[cryptography/cryptography.h](#), 1357

[cryptography/ecies.c](#), 1393
[deprecated_ecies_key_public_bin](#)
[cryptography/cryptography.h](#), 1357
[cryptography/ecies.c](#), 1393
[deprecated_ecies_key_public_hex](#)
[cryptography/cryptography.h](#), 1357
[cryptography/ecies.c](#), 1394
[deprecated_ecies_start](#)
[cryptography/cryptography.h](#), 1358
[cryptography/ecies.c](#), 1394
[deprecated_ecies_stop](#)
[cryptography/cryptography.h](#), 1358
[cryptography/ecies.c](#), 1394
[deprecated_hmac_digest](#)
[cryptography/cryptography.h](#), 1358
[cryptography/hmac.c](#), 1404
[deprecated_hmac_md4](#)
[cryptography/cryptography.h](#), 1358
[cryptography/hmac.c](#), 1404
[deprecated_hmac_md5](#)
[cryptography/cryptography.h](#), 1359
[cryptography/hmac.c](#), 1404
[deprecated_hmac_ripemd160](#)
[cryptography/cryptography.h](#), 1359
[cryptography/hmac.c](#), 1404
[deprecated_hmac_sha](#)
[cryptography/cryptography.h](#), 1359
[cryptography/hmac.c](#), 1405
[deprecated_hmac_sha1](#)
[cryptography/cryptography.h](#), 1359
[cryptography/hmac.c](#), 1405
[deprecated_hmac_sha224](#)
[cryptography/cryptography.h](#), 1359
[cryptography/hmac.c](#), 1405
[deprecated_hmac_sha256](#)
[cryptography/cryptography.h](#), 1359
[cryptography/hmac.c](#), 1405
[deprecated_hmac_sha384](#)
[cryptography/cryptography.h](#), 1359
[cryptography/hmac.c](#), 1405
[deprecated_hmac_sha512](#)
[cryptography/cryptography.h](#), 1359
[cryptography/hmac.c](#), 1405
[deprecated_scramble_alloc](#)
[cryptography/cryptography.h](#), 1359
[cryptography/scramble.c](#), 1422
[deprecated_scramble_body_data](#)
[cryptography/cryptography.h](#), 1360
[cryptography/scramble.c](#), 1423
[deprecated_scramble_body_hash](#)
[cryptography/cryptography.h](#), 1360
[cryptography/scramble.c](#), 1423
[deprecated_scramble_body_length](#)
[cryptography/cryptography.h](#), 1360
[cryptography/scramble.c](#), 1423
[deprecated_scramble_cleanup](#)
[cryptography/cryptography.h](#), 1361

- cryptography/scramble.c, [1423](#)
- deprecated_scramble_decrypt
 - cryptography/cryptography.h, [1361](#)
 - cryptography/scramble.c, [1424](#)
- deprecated_scramble_encrypt
 - cryptography/cryptography.h, [1361](#)
 - cryptography/scramble.c, [1424](#)
- deprecated_scramble_free
 - cryptography/cryptography.h, [1362](#)
 - cryptography/scramble.c, [1424](#)
- deprecated_scramble_import
 - cryptography/cryptography.h, [1362](#)
 - cryptography/scramble.c, [1425](#)
- deprecated_scramble_orig_length
 - cryptography/cryptography.h, [1362](#)
 - cryptography/scramble.c, [1425](#)
- deprecated_scramble_total_length
 - cryptography/cryptography.h, [1363](#)
 - cryptography/scramble.c, [1425](#)
- deprecated_scramble_vector_data
 - cryptography/cryptography.h, [1363](#)
 - cryptography/scramble.c, [1426](#)
- deprecated_scramble_vector_length
 - cryptography/cryptography.h, [1363](#)
 - cryptography/scramble.c, [1426](#)
- deprecated_symmetric_decrypt
 - cryptography/cryptography.h, [1363](#)
 - cryptography/symmetric.c, [1432](#)
- deprecated_symmetric_encrypt
 - cryptography/cryptography.h, [1364](#)
 - cryptography/symmetric.c, [1432](#)
- deprecated_symmetric_key
 - cryptography/cryptography.h, [1364](#)
 - cryptography/symmetric.c, [1432](#)
- deprecated_symmetric_vector
 - cryptography/cryptography.h, [1364](#)
 - cryptography/symmetric.c, [1433](#)
- dequeue
 - controller.h, [777](#)
 - queue.c, [783](#)
- derived
 - engine/status/statistics.c, [805](#)
- derror_t, [36](#)
 - code, [36](#)
 - message, [36](#)
- description
 - cache_keys_t, [29](#)
 - cached_store_t, [34](#)
 - dmime_chunk_key_t, [41](#)
 - magma_keys_t, [86](#)
 - relay_keys_t, [125](#)
 - server_keys_t, [129](#)
 - signet_field_key_t, [134](#)
- deserialization.c
 - deserialize_count_digits, [1321](#)
 - deserialize_int16, [1322](#)
 - deserialize_int32, [1322](#)
 - deserialize_int64, [1322](#)
 - deserialize_ns, [1322](#)
 - deserialize_ssz, [1323](#)
 - deserialize_st, [1323](#)
 - deserialize_sz, [1324](#)
 - deserialize_uint16, [1324](#)
 - deserialize_uint32, [1324](#)
 - deserialize_uint64, [1324](#)
- deserialize
 - cached_store_t, [34](#)
- deserialize_count_digits
 - consumers.h, [1313](#)
 - deserialization.c, [1321](#)
- deserialize_ec_privkey
 - crypto_pub.c, [1511](#)
- deserialize_ec_pubkey
 - crypto_pub.c, [1511](#)
- deserialize_ed25519_privkey
 - crypto_pub.c, [1511](#)
- deserialize_ed25519_pubkey
 - crypto_pub.c, [1511](#)
- deserialize_int16
 - consumers.h, [1313](#)
 - deserialization.c, [1322](#)
- deserialize_int32
 - consumers.h, [1313](#)
 - deserialization.c, [1322](#)
- deserialize_int64
 - consumers.h, [1313](#)
 - deserialization.c, [1322](#)
- deserialize_ns
 - consumers.h, [1313](#)
 - deserialization.c, [1322](#)
- deserialize_ssz
 - consumers.h, [1314](#)
 - deserialization.c, [1323](#)
- deserialize_st
 - consumers.h, [1314](#)
 - deserialization.c, [1323](#)
- deserialize_sz
 - consumers.h, [1315](#)
 - deserialization.c, [1324](#)
- deserialize_uint16
 - consumers.h, [1315](#)
 - deserialization.c, [1324](#)
- deserialize_uint32
 - consumers.h, [1315](#)
 - deserialization.c, [1324](#)
- deserialize_uint64
 - consumers.h, [1315](#)
 - deserialization.c, [1324](#)
- dest_keyslot
 - dmime_chunk_key_t, [41](#)
- dest_orig
 - dmime_envelope_object_t, [44](#)
- dest_orig_fp
 - dmime_envelope_object_t, [44](#)

- destination
 - dmime_message_t, [46](#)
 - dmime_object_t, [48](#)
- destroy_cache_entry
 - cache_pub.c, [1709](#)
- destroy_dime_record
 - mrec_pub.c, [1768](#)
- destructor
 - cached_store_t, [35](#)
- dh2048
 - openssl.c, [1412](#)
 - parameters.c, [1416](#)
- dh4096
 - openssl.c, [1412](#)
 - parameters.c, [1416](#)
- DH_check_d
 - symbols.c, [1988](#)
 - symbols.h, [2028](#)
- dh_exchange_2048
 - cryptography/cryptography.h, [1365](#)
 - parameters.c, [1414](#)
- dh_exchange_4096
 - cryptography/cryptography.h, [1365](#)
 - parameters.c, [1414](#)
- DH_free_d
 - symbols.c, [1988](#)
 - symbols.h, [2028](#)
- DH_generate_parameters_ex_d
 - symbols.h, [2028](#)
- DH_new_d
 - symbols.c, [1988](#)
 - symbols.h, [2028](#)
- dh_params_2048
 - cryptography/cryptography.h, [1366](#)
 - parameters.c, [1415](#)
- dh_params_4096
 - cryptography/cryptography.h, [1366](#)
 - parameters.c, [1415](#)
- dh_params_generate
 - cryptography/cryptography.h, [1366](#)
 - parameters.c, [1415](#)
- dh_params_generate_callback
 - cryptography/cryptography.h, [1366](#)
 - parameters.c, [1415](#)
- dh_static_2048
 - cryptography/cryptography.h, [1366](#)
 - parameters.c, [1415](#)
- dh_static_4096
 - cryptography/cryptography.h, [1366](#)
 - parameters.c, [1416](#)
- dhparam_lock
 - openssl.c, [1412](#)
 - parameters.c, [1416](#)
- dhparams_large_keys
 - magma_t, [93](#)
- dhparams_rotate
 - magma_t, [93](#)
- digest
 - ds, [59](#)
- digest.c
 - digest_id, [1389](#)
 - digest_length_output, [1389](#)
 - digest_name, [1389](#)
- digest_id
 - cryptography/cryptography.h, [1366](#)
 - digest.c, [1389](#)
- digest_length_output
 - digest.c, [1389](#)
- digest_name
 - cryptography/cryptography.h, [1367](#)
 - digest.c, [1389](#)
- digest_t
 - cryptography/cryptography.h, [1349](#)
- digest_type
 - ds, [59](#)
- digits.c
 - int16_digits, [404](#)
 - int32_digits, [404](#)
 - int64_digits, [405](#)
 - int8_digits, [405](#)
 - uint16_digits, [405](#)
 - uint32_digits, [405](#)
 - uint64_digits, [405](#)
 - uint8_digits, [406](#)
- diglen
 - ds, [59](#)
- dime
 - magma_t, [93](#)
- dime/ed25519/ed25519.c
 - curved25519_scalarmult_basepoint, [1665](#)
 - ed25519_publickey, [1665](#)
 - ed25519_sign, [1665](#)
 - ed25519_sign_open, [1665](#)
- dime_ctx.h
 - LOG_CODE_DEBUG, [1567](#)
 - LOG_CODE_ERROR, [1567](#)
 - LOG_CODE_INFO, [1567](#)
- DIME_ENCRYPTED_MSG
 - signet/common.h, [1576](#)
- DIME_ENCRYPTED_ORG_KEYS
 - signet/common.h, [1576](#)
- DIME_ENCRYPTED_USER_KEYS
 - signet/common.h, [1576](#)
- DIME_MSG_TRACING
 - signet/common.h, [1576](#)
- DIME_ORG_KEYS
 - signet/common.h, [1576](#)
- DIME_ORG_SIGNET
 - signet/common.h, [1576](#)
- DIME_SSR
 - signet/common.h, [1576](#)
- DIME_USER_KEYS
 - signet/common.h, [1576](#)
- DIME_USER_SIGNET

- signet/common.h, 1576
- dime_ctx, 37
 - log_callback, 37
- dime_ctx.c
 - dime_ctx_free, 1564
 - dime_ctx_log, 1564
 - dime_ctx_new, 1564
 - LOG_LEVEL_DEBUG, 1564
 - LOG_LEVEL_ERROR, 1564
 - LOG_LEVEL_INFO, 1564
- dime_ctx.h
 - dime_ctx_free, 1567
 - dime_ctx_log, 1567
 - dime_ctx_new, 1568
 - dime_ctx_t, 1567
 - DIME_LOG_DEBUG, 1566
 - DIME_LOG_ERROR, 1566
 - DIME_LOG_INFO, 1567
 - log_code_t, 1567
 - log_function_t, 1567
 - LOG_LEVEL_DEBUG, 1568
 - LOG_LEVEL_ERROR, 1568
 - LOG_LEVEL_INFO, 1568
- dime_ctx_free
 - dime_ctx.c, 1564
 - dime_ctx.h, 1567
- dime_ctx_log
 - dime_ctx.c, 1564
 - dime_ctx.h, 1567
- dime_ctx_new
 - dime_ctx.c, 1564
 - dime_ctx.h, 1568
- dime_ctx_t
 - dime_ctx.h, 1567
- dime_dmsg_actor_to_string
 - crypto.h, 1583
 - dmsg.c, 1592
- dime_dmsg_chunks_sig_origin_sign
 - crypto.h, 1583
 - dmsg.c, 1592
- dime_dmsg_kek_in_derive
 - crypto.h, 1584
 - dmsg.c, 1592
- dime_dmsg_message_binary_deserialize
 - crypto.h, 1584
 - dmsg.c, 1593
- dime_dmsg_message_binary_serialize
 - crypto.h, 1584
 - dmsg.c, 1593
- dime_dmsg_message_decrypt_as_auth
 - crypto.h, 1585
 - dmsg.c, 1593
- dime_dmsg_message_decrypt_as_dest
 - crypto.h, 1585
 - dmsg.c, 1594
- dime_dmsg_message_decrypt_as_orig
 - crypto.h, 1585
- dmsg.c, 1594
- dime_dmsg_message_decrypt_as_recip
 - crypto.h, 1586
 - dmsg.c, 1594
- dime_dmsg_message_destroy
 - crypto.h, 1586
 - dmsg.c, 1595
- dime_dmsg_message_encrypt
 - crypto.h, 1586
 - dmsg.c, 1595
- dime_dmsg_message_envelope_decrypt
 - crypto.h, 1586
 - dmsg.c, 1595
- dime_dmsg_message_state_get
 - crypto.h, 1587
 - dmsg.c, 1595
- dime_dmsg_object_chunk_create
 - crypto.h, 1587
 - dmsg.c, 1596
- dime_dmsg_object_chunklist_destroy
 - crypto.h, 1587
 - dmsg.c, 1596
- dime_dmsg_object_destroy
 - crypto.h, 1587
 - dmsg.c, 1596
- dime_dmsg_object_dump
 - crypto.h, 1588
 - dmsg.c, 1596
- dime_dmsg_object_state_init
 - crypto.h, 1588
 - dmsg.c, 1597
- dime_dmsg_object_state_to_string
 - crypto.h, 1588
 - dmsg.c, 1597
- dime_dmtip_command_create
 - providers/dime/dmtip/commands.c, 1607
 - providers/dime/dmtip/commands.h, 1625
- dime_dmtip_command_data
 - providers/dime/dmtip/commands.c, 1608
 - providers/dime/dmtip/commands.h, 1625
- dime_dmtip_command_destroy
 - providers/dime/dmtip/commands.c, 1608
 - providers/dime/dmtip/commands.h, 1625
- dime_dmtip_command_ehlo
 - providers/dime/dmtip/commands.c, 1608
 - providers/dime/dmtip/commands.h, 1626
- dime_dmtip_command_format
 - providers/dime/dmtip/commands.c, 1608
 - providers/dime/dmtip/commands.h, 1626
- dime_dmtip_command_helo
 - providers/dime/dmtip/commands.c, 1608
 - providers/dime/dmtip/commands.h, 1626
- dime_dmtip_command_help
 - providers/dime/dmtip/commands.c, 1609
 - providers/dime/dmtip/commands.h, 1626
- dime_dmtip_command_hist
 - providers/dime/dmtip/commands.c, 1609

- providers/dime/dmtp/commands.h, 1627
- dime_dmtp_command_mail
 - providers/dime/dmtp/commands.c, 1609
 - providers/dime/dmtp/commands.h, 1627
- dime_dmtp_command_mode
 - providers/dime/dmtp/commands.c, 1610
 - providers/dime/dmtp/commands.h, 1627
- dime_dmtp_command_noop
 - providers/dime/dmtp/commands.c, 1610
 - providers/dime/dmtp/commands.h, 1627
- dime_dmtp_command_parse
 - providers/dime/dmtp/commands.c, 1610
 - providers/dime/dmtp/commands.h, 1628
- dime_dmtp_command_quit
 - providers/dime/dmtp/commands.c, 1610
 - providers/dime/dmtp/commands.h, 1628
- dime_dmtp_command_rcpt
 - providers/dime/dmtp/commands.c, 1611
 - providers/dime/dmtp/commands.h, 1628
- dime_dmtp_command_rset
 - providers/dime/dmtp/commands.c, 1611
 - providers/dime/dmtp/commands.h, 1628
- dime_dmtp_command_sgnt_domain
 - providers/dime/dmtp/commands.c, 1611
 - providers/dime/dmtp/commands.h, 1629
- dime_dmtp_command_sgnt_user
 - providers/dime/dmtp/commands.c, 1611
 - providers/dime/dmtp/commands.h, 1629
- dime_dmtp_command_starttls
 - providers/dime/dmtp/commands.c, 1612
 - providers/dime/dmtp/commands.h, 1629
- dime_dmtp_command_vrfy_domain
 - providers/dime/dmtp/commands.c, 1612
 - providers/dime/dmtp/commands.h, 1630
- dime_dmtp_command_vrfy_user
 - providers/dime/dmtp/commands.c, 1612
 - providers/dime/dmtp/commands.h, 1630
- dime_errcode_t
 - error_codes.h, 1691
- dime_keys_enckey_fetch
 - providers/dime/signet/keys.c, 585
 - providers/dime/signet/keys.h, 667
- dime_keys_file_create
 - providers/dime/signet/keys.c, 585
 - providers/dime/signet/keys.h, 667
- dime_keys_generate
 - providers/dime/signet/keys.c, 586
 - providers/dime/signet/keys.h, 668
- dime_keys_signkey_fetch
 - providers/dime/signet/keys.c, 586
 - providers/dime/signet/keys.h, 668
- DIME_LOG_DEBUG
 - dime_ctx.h, 1566
- DIME_LOG_ERROR
 - dime_ctx.h, 1566
- DIME_LOG_INFO
 - dime_ctx.h, 1567
- dime_msg_policy
 - mrec.h, 1765
- dime_num
 - dmime_message_t, 46
- DIME_NUMBER_SIZE
 - signet/common.h, 1574
- dime_number_t
 - signet/common.h, 1576
- dime_number_to_str
 - general.c, 1786
 - signet/common.h, 1579
- dime_prsr_envelope_destroy
 - parse.h, 1600
 - parser.c, 1603
- dime_prsr_envelope_format
 - parse.h, 1600
 - parser.c, 1603
- dime_prsr_envelope_parse
 - parse.h, 1600
 - parser.c, 1604
- dime_prsr_headers_create
 - parse.h, 1601
 - parser.c, 1604
- dime_prsr_headers_destroy
 - parse.h, 1601
 - parser.c, 1604
- dime_prsr_headers_format
 - parse.h, 1601
 - parser.c, 1604
- dime_prsr_headers_parse
 - parse.h, 1601
 - parser.c, 1605
- DIME_RECORD_DNS_PREFIX
 - mrec.h, 1765
- dime_record_t, 38
 - dx, 38
 - expiry, 38
 - policy, 39
 - pubkey, 39
 - subdomain, 39
 - syndicates, 39
 - tlssig, 39
 - validated, 39
 - version, 39
- dime_sgnt_enckey_fetch
 - signet.c, 1791
 - signet.h, 1809
- dime_sgnt_enckey_set
 - signet.c, 1791
 - signet.h, 1810
- dime_sgnt_fid_count_get
 - signet.c, 1791
 - signet.h, 1810
- dime_sgnt_fid_exists
 - signet.c, 1791
 - signet.h, 1810
- dime_sgnt_fid_num_fetch

signet.c, [1792](#)
 signet.h, [1810](#)
dime_sgnt_fid_num_remove
 signet.c, [1792](#)
 signet.h, [1811](#)
dime_sgnt_field_defined_create
 signet.c, [1792](#)
 signet.h, [1811](#)
dime_sgnt_field_defined_set
 signet.c, [1793](#)
 signet.h, [1811](#)
dime_sgnt_field_undefined_create
 signet.c, [1793](#)
 signet.h, [1812](#)
dime_sgnt_field_undefined_fetch
 signet.c, [1793](#)
 signet.h, [1812](#)
dime_sgnt_field_undefined_remove
 signet.c, [1794](#)
 signet.h, [1812](#)
dime_sgnt_file_create
 signet.c, [1794](#)
 signet.h, [1813](#)
dime_sgnt_fingerprint_crypto
 signet.c, [1794](#)
 signet.h, [1813](#)
dime_sgnt_fingerprint_full
 signet.c, [1795](#)
 signet.h, [1813](#)
dime_sgnt_fingerprint_id
 signet.c, [1795](#)
 signet.h, [1814](#)
dime_sgnt_fingerprint_ssr
 signet.c, [1795](#)
 signet.h, [1814](#)
dime_sgnt_id_fetch
 signet.c, [1796](#)
 signet.h, [1814](#)
dime_sgnt_id_set
 signet.c, [1796](#)
 signet.h, [1815](#)
dime_sgnt_msg_sig_verify
 signet.c, [1796](#)
 signet.h, [1815](#)
dime_sgnt_sig_coc_sign
 signet.c, [1797](#)
 signet.h, [1815](#)
dime_sgnt_sig_crypto_sign
 signet.c, [1797](#)
 signet.h, [1815](#)
dime_sgnt_sig_full_sign
 signet.c, [1797](#)
 signet.h, [1816](#)
dime_sgnt_sig_id_sign
 signet.c, [1797](#)
 signet.h, [1816](#)
dime_sgnt_sig_ssr_sign
 signet.c, [1798](#)
 signet.h, [1816](#)
dime_sgnt_signet_b64_deserialize
 signet.c, [1798](#)
 signet.h, [1817](#)
dime_sgnt_signet_b64_serialize
 signet.c, [1798](#)
 signet.h, [1817](#)
dime_sgnt_signet_binary_deserialize
 signet.c, [1799](#)
 signet.h, [1817](#)
dime_sgnt_signet_binary_serialize
 signet.c, [1799](#)
 signet.h, [1817](#)
dime_sgnt_signet_create
 signet.c, [1799](#)
 signet.h, [1818](#)
dime_sgnt_signet_create_w_keys
 signet.c, [1799](#)
 signet.h, [1818](#)
dime_sgnt_signet_crypto_split
 signet.c, [1800](#)
 signet.h, [1818](#)
dime_sgnt_signet_destroy
 signet.c, [1800](#)
 signet.h, [1819](#)
dime_sgnt_signet_dump
 signet.c, [1800](#)
 signet.h, [1819](#)
dime_sgnt_signet_dupe
 signet.c, [1800](#)
 signet.h, [1819](#)
dime_sgnt_signet_full_split
 signet.c, [1801](#)
 signet.h, [1819](#)
dime_sgnt_signet_load
 signet.c, [1801](#)
 signet.h, [1820](#)
dime_sgnt_signkey_fetch
 signet.c, [1801](#)
 signet.h, [1820](#)
dime_sgnt_signkey_set
 signet.c, [1801](#)
 signet.h, [1820](#)
dime_sgnt_signkeys_msg_fetch
 signet.c, [1802](#)
 signet.h, [1820](#)
dime_sgnt_signkeys_signet_fetch
 signet.c, [1802](#)
 signet.h, [1821](#)
dime_sgnt_signkeys_software_fetch
 signet.c, [1802](#)
 signet.h, [1821](#)
dime_sgnt_signkeys_tls_fetch
 signet.c, [1803](#)
 signet.h, [1821](#)
dime_sgnt_sok_create

- signet.c, 1803
- signet.h, 1822
- dime_sgnt_sok_num_fetch
 - signet.c, 1803
 - signet.h, 1822
- dime_sgnt_state_to_str
 - signet.c, 1804
 - signet.h, 1822
- dime_sgnt_type_get
 - signet.c, 1804
 - signet.h, 1823
- dime_sgnt_type_set
 - signet.c, 1804
 - signet.h, 1823
- dime_sgnt_validate_all
 - signet.c, 1805
 - signet.h, 1823
- dime_sub_policy
 - mrec.h, 1765
- DIME_VERSION_NO
 - mrec.h, 1765
- DIRECT
 - checkers.h, 1262
- display
 - dmime_message_t, 47
 - dmime_object_t, 48
- DISPLAY_BOUNCE
 - dmessage/common.h, 1571
- display_usage
 - args.c, 760
 - context.h, 763
- disposition
 - teacher_data_t, 157
- dkim
 - magma_t, 93
 - smtp_inbound_prefs_t, 142
 - smtp_session_t, 152
- dkim.c
 - dkim_engine, 1279
 - dkim_memory_alloc, 1277
 - dkim_memory_free, 1277
 - DKIM_PROCESS_ALL, 1276
 - dkim_signature_create, 1277
 - dkim_signature_verify, 1278
 - dkim_start, 1278
 - dkim_stop, 1278
 - dkim_version, 1279
 - lib_load_dkim, 1279
 - lib_version_dkim, 1279
- dkim_body_d
 - symbols.c, 1988
 - symbols.h, 2028
- dkim_chunk_d
 - symbols.c, 1988
 - symbols.h, 2028
- dkim_close_d
 - symbols.c, 1988
- symbols.h, 2028
- dkim_engine
 - dkim.c, 1279
- dkim_eoh_d
 - symbols.c, 1988
 - symbols.h, 2028
- dkim_eom_d
 - symbols.c, 1988
 - symbols.h, 2029
- dkim_free_d
 - symbols.c, 1988
 - symbols.h, 2029
- dkim_geterror_d
 - symbols.c, 1989
 - symbols.h, 2029
- dkim_getresultstr_d
 - symbols.c, 1989
 - symbols.h, 2029
- dkim_getsighdrx_d
 - symbols.c, 1989
 - symbols.h, 2029
- dkim_header_d
 - symbols.c, 1989
 - symbols.h, 2029
- dkim_init_d
 - symbols.c, 1989
 - symbols.h, 2029
- dkim_libversion_d
 - symbols.c, 1989
 - symbols.h, 2029
- dkim_memory_alloc
 - checkers.h, 1262
 - dkim.c, 1277
- dkim_memory_free
 - checkers.h, 1262
 - dkim.c, 1277
- dkim_mfree_d
 - symbols.c, 1989
 - symbols.h, 2029
- DKIM_PROCESS_ALL
 - dkim.c, 1276
- dkim_sign_d
 - symbols.c, 1989
 - symbols.h, 2029
- dkim_signature_create
 - checkers.h, 1263
 - dkim.c, 1277
- dkim_signature_verify
 - checkers.h, 1263
 - dkim.c, 1278
- dkim_start
 - checkers.h, 1264
 - dkim.c, 1278
- dkim_stop
 - checkers.h, 1264
 - dkim.c, 1278
- dkim_test_dns_put_d

- symbols.c, 1989
- symbols.h, 2029
- dkim_verify_d
 - symbols.c, 1989
 - symbols.h, 2029
- dkim_version
 - dkim.c, 1279
- dkimaction
 - smtp_inbound_prefs_t, 142
- dmessage/common.h
 - ALTERNATE_PADDING_ALGORITHM_ENABLED, 1569
 - ALTERNATE_USER_KEY_APPLIED_TO_DATE, 1569
 - CHUNK_SECTION_ATTACH, 1571
 - CHUNK_SECTION_DISPLAY, 1571
 - CHUNK_SECTION_ENVELOPE, 1571
 - CHUNK_SECTION_METADATA, 1571
 - CHUNK_SECTION_NONE, 1571
 - CHUNK_SECTION_SIG, 1571
 - CHUNK_TYPE_ALTERNATE, 1571
 - CHUNK_TYPE_ATTACH_CONTENT, 1571
 - CHUNK_TYPE_DESTINATION, 1571
 - CHUNK_TYPE_DISPLAY_CONTENT, 1571
 - CHUNK_TYPE_EPHEMERAL, 1571
 - CHUNK_TYPE_META_COMMON, 1571
 - CHUNK_TYPE_META_OTHER, 1571
 - CHUNK_TYPE_NONE, 1571
 - CHUNK_TYPE_ORIGIN, 1571
 - CHUNK_TYPE_SIG_AUTHOR_FULL, 1571
 - CHUNK_TYPE_SIG_AUTHOR_TREE, 1571
 - CHUNK_TYPE_SIG_ORIGIN_DISPLAY_BOUNCE, 1571
 - CHUNK_TYPE_SIG_ORIGIN_FULL, 1571
 - CHUNK_TYPE_SIG_ORIGIN_META_BOUNCE, 1571
 - CHUNK_HEADER_SIZE, 1570
 - CHUNK_LENGTH_SIZE, 1570
 - DATA_SEGMENT_CONTINUATION_ENABLED, 1570
 - DEFAULT_CHUNK_FLAGS, 1570
 - DISPLAY_BOUNCE, 1571
 - dmime_bounce_type_t, 1571
 - dmime_chunk_keys, 1572
 - dmime_chunk_section_t, 1571
 - DMIME_CHUNK_TYPE_MAX, 1570
 - dmime_chunk_type_t, 1571
 - DMIME_NUM_COMMON_HEADERS, 1570
 - dmime_payload_type_t, 1571
 - GZIP_COMPRESSION_ENABLED, 1570
 - MESSAGE_HEADER_SIZE, 1570
 - MESSAGE_LENGTH_SIZE, 1570
 - META_BOUNCE, 1571
 - MINIMUM_PAYLOAD_SIZE, 1570
 - PAYLOAD_TYPE_EPHEMERAL, 1572
 - PAYLOAD_TYPE_NONE, 1572
 - PAYLOAD_TYPE_SIGNATURE, 1572
 - PAYLOAD_TYPE_STANDARD, 1572
 - TRACING_HEADER_SIZE, 1570
- DMIME_OBJECT_STATE_COMPLETE
 - crypto.h, 1583
- DMIME_OBJECT_STATE_CREATION
 - crypto.h, 1583
- DMIME_OBJECT_STATE_INCOMPLETE_ENVELOPE
 - crypto.h, 1583
- DMIME_OBJECT_STATE_INCOMPLETE_METADATA
 - crypto.h, 1583
- DMIME_OBJECT_STATE_LOADED_ENVELOPE
 - crypto.h, 1583
- DMIME_OBJECT_STATE_LOADED_SIGNETS
 - crypto.h, 1583
- DMIME_OBJECT_STATE_NONE
 - crypto.h, 1583
- dmime_actor_t
 - crypto.h, 1582
- dmime_bounce_type_t
 - dmessage/common.h, 1571
- dmime_chunk_key_t, 41
 - auth_keyslot, 41
 - description, 41
 - dest_keyslot, 41
 - encrypted, 41
 - name, 41
 - orig_keyslot, 41
 - payload, 41
 - recp_keyslot, 42
 - required, 42
 - section, 42
 - sequential, 42
 - unique, 42
- dmime_chunk_keys
 - dmessage/common.h, 1572
 - dmsg.c, 1597
- dmime_chunk_section_t
 - dmessage/common.h, 1571
- DMIME_CHUNK_TYPE_MAX
 - dmessage/common.h, 1570
- dmime_chunk_type_t
 - dmessage/common.h, 1571
- dmime_common_headers_t, 43
 - headers, 43
- dmime_encrypted_payload_t
 - dmsg.c, 1591
- dmime_envelope_object_t, 44
 - auth_recip, 44
 - auth_recip_fp, 44
 - dest_orig, 44
 - dest_orig_fp, 44
- dmime_ephemeral_payload_t
 - dmsg.c, 1591
- dmime_header_key_t, 45
 - label, 45
 - label_length, 45
 - required, 45
- dmime_header_keys
 - parse.h, 1602
 - parser.c, 1605
- dmime_header_type_t
 - parse.h, 1599

- dmime_kek_t
 - crypto.h, 1589
- dmime_kekset_t
 - dmsg.c, 1592
- dmime_keyslot_t
 - dmsg.c, 1597
- dmime_message_chunk_state_t
 - crypto.h, 1582
- dmime_message_chunk_t
 - crypto.h, 1589
- dmime_message_state_t
 - crypto.h, 1582
- dmime_message_t, 46
 - attach, 46
 - author_full_sig, 46
 - author_tree_sig, 46
 - common_headers, 46
 - destination, 46
 - dime_num, 46
 - display, 47
 - ephemeral, 47
 - origin, 47
 - origin_display_bounce_sig, 47
 - origin_full_sig, 47
 - origin_meta_bounce_sig, 47
 - other_headers, 47
 - size, 47
 - state, 47
 - tracing, 47
- DMIME_NUM_COMMON_HEADERS
 - dmessage/common.h, 1570
- dmime_object_chunk_t
 - crypto.h, 1582
- dmime_object_state_t
 - crypto.h, 1583
- dmime_object_t, 48
 - actor, 48
 - attach, 48
 - author, 48
 - common_headers, 48
 - destination, 48
 - display, 48
 - fp_author, 49
 - fp_destination, 49
 - fp_origin, 49
 - fp_recipient, 49
 - origin, 49
 - other_headers, 49
 - recipient, 49
 - signet_author, 49
 - signet_destination, 49
 - signet_origin, 49
 - signet_recipient, 49
 - state, 49
- dmime_payload_type_t
 - dmessage/common.h, 1571
- dmime_signature_payload_t
 - dmsg.c, 1592
- dmime_standard_payload_t
 - dmsg.c, 1597
- dmime_tracing_t
 - crypto.h, 1589
- dmsg.c
 - __attribute__, 1592
 - CKEY_EMPTY, 1591
 - dime_dmsg_actor_to_string, 1592
 - dime_dmsg_chunks_sig_origin_sign, 1592
 - dime_dmsg_kek_in_derive, 1592
 - dime_dmsg_message_binary_deserialize, 1593
 - dime_dmsg_message_binary_serialize, 1593
 - dime_dmsg_message_decrypt_as_auth, 1593
 - dime_dmsg_message_decrypt_as_dest, 1594
 - dime_dmsg_message_decrypt_as_orig, 1594
 - dime_dmsg_message_decrypt_as_recip, 1594
 - dime_dmsg_message_destroy, 1595
 - dime_dmsg_message_encrypt, 1595
 - dime_dmsg_message_envelope_decrypt, 1595
 - dime_dmsg_message_state_get, 1595
 - dime_dmsg_object_chunk_create, 1596
 - dime_dmsg_object_chunklist_destroy, 1596
 - dime_dmsg_object_destroy, 1596
 - dime_dmsg_object_dump, 1596
 - dime_dmsg_object_state_init, 1597
 - dime_dmsg_object_state_to_string, 1597
 - dmime_chunk_keys, 1597
 - dmime_encrypted_payload_t, 1591
 - dmime_ephemeral_payload_t, 1591
 - dmime_kekset_t, 1592
 - dmime_keyslot_t, 1597
 - dmime_signature_payload_t, 1592
 - dmime_standard_payload_t, 1597
- DMTP
 - engine/config/servers/servers.h, 753
- dmtplib/session.c
 - dmtplib_session_destroy, 2100
 - dmtplib_session_reset, 2100
- DMTP_ARG_ANGULAR_BRCKT
 - providers/dime/dmtplib/commands.h, 1624
- DMTP_ARG_NONE
 - providers/dime/dmtplib/commands.h, 1624
- DMTP_ARG_PLAIN
 - providers/dime/dmtplib/commands.h, 1624
- DMTP_ARG_SQUARE_BRCKT
 - providers/dime/dmtplib/commands.h, 1624
- DMTP_COMMAND_INVALID
 - providers/dime/dmtplib/commands.h, 1624
- DMTP_DATA
 - providers/dime/dmtplib/commands.h, 1624
- DMTP_EHLO
 - providers/dime/dmtplib/commands.h, 1624
- DMTP_HELO
 - providers/dime/dmtplib/commands.h, 1624
- DMTP_HELP
 - providers/dime/dmtplib/commands.h, 1624

- DMTP_HIST
 - providers/dime/dmtp/commands.h, 1624
- DMTP_MAIL
 - providers/dime/dmtp/commands.h, 1624
- DMTP_MODE
 - providers/dime/dmtp/commands.h, 1624
- DMTP_MODE_DMTP
 - providers/dime/dmtp/commands.h, 1625
 - providers/dime/signet-resolver/dmtp.h, 838
- dmtp_mode_dmtp
 - providers/dime/dmtp/commands.h, 1625
 - providers/dime/signet-resolver/dmtp.h, 838
- dmtp_mode_dual
 - providers/dime/dmtp/commands.h, 1625
 - providers/dime/signet-resolver/dmtp.h, 838
- dmtp_mode_esmtp
 - providers/dime/dmtp/commands.h, 1625
 - providers/dime/signet-resolver/dmtp.h, 838
- DMTP_MODE_NONE
 - providers/dime/dmtp/commands.h, 1625
 - providers/dime/signet-resolver/dmtp.h, 838
- DMTP_MODE_SMTP
 - providers/dime/dmtp/commands.h, 1625
 - providers/dime/signet-resolver/dmtp.h, 838
- dmtp_mode_smtp
 - providers/dime/dmtp/commands.h, 1625
 - providers/dime/signet-resolver/dmtp.h, 838
- dmtp_mode_unknown
 - providers/dime/dmtp/commands.h, 1625
 - providers/dime/signet-resolver/dmtp.h, 838
- DMTP_NOOP
 - providers/dime/dmtp/commands.h, 1624
- DMTP_PARSE_ARGUMENT_ERROR
 - providers/dime/dmtp/commands.h, 1625
- DMTP_PARSE_COMMAND_ERROR
 - providers/dime/dmtp/commands.h, 1625
- DMTP_PARSE_INTERNAL_ERROR
 - providers/dime/dmtp/commands.h, 1625
- DMTP_PARSE_INVALID_CALL
 - providers/dime/dmtp/commands.h, 1625
- DMTP_PARSE_SUCCESS
 - providers/dime/dmtp/commands.h, 1625
- DMTP_QUIT
 - providers/dime/dmtp/commands.h, 1624
- DMTP_RCPT
 - providers/dime/dmtp/commands.h, 1624
- DMTP_RSET
 - providers/dime/dmtp/commands.h, 1624
- DMTP_SGNT
 - providers/dime/dmtp/commands.h, 1624
- DMTP_STARTTLS
 - providers/dime/dmtp/commands.h, 1624
- DMTP_VRFY
 - providers/dime/dmtp/commands.h, 1624
- dmtp_argument_t, 51
 - arg_name, 51
 - arg_name_len, 51
 - size, 51
 - type, 51
- dmtp_argument_type_t
 - providers/dime/dmtp/commands.h, 1624
- dmtp_command_key_t, 52
 - args, 52
 - com_name, 52
 - com_name_len, 52
- dmtp_command_list
 - providers/dime/dmtp/commands.c, 1613
 - providers/dime/dmtp/commands.h, 1630
- dmtp_command_t, 53
 - args, 53
 - type, 53
- dmtp_command_type_t
 - providers/dime/dmtp/commands.h, 1624
- dmtp_commands
 - servers/dmtp/commands.h, 1631
- DMTP_COMMANDS_NUM
 - providers/dime/dmtp/commands.h, 1624
- dmtp_compare
 - servers/dmtp/commands.c, 1614
 - servers/dmtp/dmtp.h, 845
- dmtp_data
 - servers/dmtp/dmtp.c, 1722
 - servers/dmtp/dmtp.h, 845
- dmtp_ehlo
 - servers/dmtp/dmtp.c, 1723
 - servers/dmtp/dmtp.h, 845
- DMTP_EMPTY_ARG
 - providers/dime/dmtp/commands.c, 1607
- dmtp_helo
 - servers/dmtp/dmtp.c, 1723
 - servers/dmtp/dmtp.h, 845
- dmtp_help
 - servers/dmtp/dmtp.c, 1723
 - servers/dmtp/dmtp.h, 845
- dmtp_hist
 - servers/dmtp/dmtp.c, 1723
 - servers/dmtp/dmtp.h, 846
- DMTP_IGNORE_ARG
 - providers/dime/dmtp/commands.c, 1607
- dmtp_init
 - servers/dmtp/dmtp.c, 1723
 - servers/dmtp/dmtp.h, 846
- dmtp_invalid
 - servers/dmtp/dmtp.c, 1724
 - servers/dmtp/dmtp.h, 846
- DMTP_LINE_BUF_SIZE
 - providers/dime/signet-resolver/dmtp.h, 837
- dmtp_mail
 - servers/dmtp/dmtp.c, 1724
 - servers/dmtp/dmtp.h, 846
- dmtp_mail_datatype_t
 - providers/dime/signet-resolver/dmtp.h, 838
- dmtp_mail_rettype_t
 - providers/dime/signet-resolver/dmtp.h, 838

- DMTP_MAX_ARGUMENT_NUM
 - providers/dime/dmtp/commands.h, 1624
- DMTP_MAX_MX_RETRIES
 - providers/dime/signet-resolver/dmtp.h, 837
- dmtp_mode
 - servers/dmtp/dmtp.c, 1724
 - servers/dmtp/dmtp.h, 846
- dmtp_mode_t
 - providers/dime/dmtp/commands.h, 1624
 - providers/dime/signet-resolver/dmtp.h, 838
- dmtp_noop
 - servers/dmtp/dmtp.c, 1724
 - servers/dmtp/dmtp.h, 847
- dmtp_parse_state_t
 - providers/dime/dmtp/commands.h, 1625
- DMTP_PORT
 - providers/dime/signet-resolver/dmtp.h, 837
- DMTP_PORT_DUAL
 - providers/dime/signet-resolver/dmtp.h, 837
- dmtp_process
 - servers/dmtp/commands.c, 1614
 - servers/dmtp/dmtp.h, 847
- dmtp_pub.c
 - dx_connect_dual, 1727
 - dx_connect_standard, 1727
 - get_signet, 1727
 - sgnt_resolv_destroy_dmtp_session, 1727
 - sgnt_resolv_dmtp_connect, 1727
 - sgnt_resolv_dmtp_data, 1728
 - sgnt_resolv_dmtp_ehlo, 1728
 - sgnt_resolv_dmtp_get_mode, 1728
 - sgnt_resolv_dmtp_get_signet, 1728
 - sgnt_resolv_dmtp_help, 1728
 - sgnt_resolv_dmtp_history, 1728
 - sgnt_resolv_dmtp_mail_from, 1728
 - sgnt_resolv_dmtp_noop, 1728
 - sgnt_resolv_dmtp_quit, 1728
 - sgnt_resolv_dmtp_rcpt_to, 1729
 - sgnt_resolv_dmtp_reset, 1729
 - sgnt_resolv_dmtp_stats, 1729
 - sgnt_resolv_dmtp_str_to_mode, 1729
 - sgnt_resolv_dmtp_verify_signet, 1729
 - verify_dx_certificate, 1729
- dmtp_quit
 - servers/dmtp/dmtp.c, 1725
 - servers/dmtp/dmtp.h, 847
- dmtp_rcpt
 - servers/dmtp/dmtp.c, 1725
 - servers/dmtp/dmtp.h, 847
- dmtp_requeue
 - servers/dmtp/commands.c, 1614
 - servers/dmtp/dmtp.h, 848
- dmtp_rset
 - servers/dmtp/dmtp.c, 1725
 - servers/dmtp/dmtp.h, 848
- dmtp_session_destroy
 - dmtp/session.c, 2100
- servers/dmtp/dmtp.h, 848
- dmtp_session_reset
 - dmtp/session.c, 2100
 - servers/dmtp/dmtp.h, 848
- dmtp_session_t, 54
 - _fd, 54
 - _inbuf, 54
 - _inpos, 54
 - active, 54
 - con, 55
 - domain, 55
 - drec, 55
 - dx, 55
 - mode, 55
- dmtp_sgnt
 - servers/dmtp/dmtp.c, 1725
 - servers/dmtp/dmtp.h, 849
- dmtp_sort
 - servers/dmtp/commands.c, 1614
 - servers/dmtp/dmtp.h, 849
- DMTP_V1_CIPHER_LIST
 - providers/dime/signet-resolver/dmtp.h, 837
- dmtp_verb
 - servers/dmtp/dmtp.c, 1726
 - servers/dmtp/dmtp.h, 849
- dmtp_vrfy
 - servers/dmtp/dmtp.c, 1726
 - servers/dmtp/dmtp.h, 849
- dns.c
 - _add_dnskey_entry, 1732
 - _add_dnskey_entry_rsa, 1733
 - _add_ds_entry, 1733
 - _clone_dnskey_record_cb, 1734
 - _compare_rdata, 1734
 - _compute_dnskey_sha_hash, 1734
 - _deserialize_dnskey_record_cb, 1735
 - _deserialize_ds_record_cb, 1735
 - _destroy_dnskey, 1735
 - _destroy_dnskey_record_cb, 1735
 - _destroy_ds, 1736
 - _destroy_ds_record_cb, 1736
 - _dnskey_domain_comparator, 1736
 - _dnskey_tag_comparator, 1736
 - _ds_comparator, 1737
 - _dump_dns_header, 1737
 - _dump_dnskey_record_cb, 1737
 - _dump_ds_record_cb, 1738
 - _fixup_dnskey_validation, 1738
 - _fixup_ds_links, 1738
 - _free_mx_records, 1738
 - _get_dnskey_by_tag, 1739
 - _get_ds_by_dnskey, 1739
 - _get_keytag, 1739
 - _get_mx_records, 1740
 - _get_rsa_dnskey, 1740
 - _get_signing_key_names, 1740
 - _get_txt_record, 1741

- [_initialize_resolver](#), 1741
- [_is_validated_key](#), 1741
- [_load_dnskey_file](#), 1742
- [_lookup_dnskey](#), 1742
- [_lookup_ds](#), 1742
- [_mem_append_canon](#), 1743
- [_rsa_verify_record](#), 1743
- [_serialize_dnskey_record_cb](#), 1744
- [_serialize_ds_record_cb](#), 1744
- [_sort_rrs_canonical](#), 1744
- [_validate_rrsig_rr](#), 1745
- [INITIALIZE_DNS](#), 1732
- [dns.h](#)
 - [__attribute__](#), 1750
 - [_add_dnskey_entry](#), 1751
 - [_add_dnskey_entry_rsa](#), 1751
 - [_add_ds_entry](#), 1751
 - [_clone_dnskey_record_cb](#), 1752
 - [_compare_rdata](#), 1752
 - [_deserialize_dnskey_record_cb](#), 1752
 - [_deserialize_ds_record_cb](#), 1753
 - [_destroy_dnskey_record_cb](#), 1753
 - [_destroy_ds_record_cb](#), 1753
 - [_dnskey_domain_comparator](#), 1754
 - [_dnskey_tag_comparator](#), 1754
 - [_ds_comparator](#), 1754
 - [_dump_dns_header](#), 1754
 - [_dump_dnskey_record_cb](#), 1755
 - [_dump_ds_record_cb](#), 1755
 - [_fixup_dnskey_validation](#), 1755
 - [_fixup_ds_links](#), 1756
 - [_get_rsa_dnskey](#), 1756
 - [_get_signing_key_names](#), 1756
 - [_initialize_resolver](#), 1756
 - [_mem_append_canon](#), 1757
 - [_serialize_dnskey_record_cb](#), 1757
 - [_serialize_ds_record_cb](#), 1757
 - [_sort_rrs_canonical](#), 1758
 - [DNSKEY_RR_FLAG_SEP](#), 1748
 - [DNSKEY_RR_FLAG_ZK](#), 1748
 - [dnskey_rr_flags_t](#), 1759
 - [DNSKEY_RR_LEN](#), 1749
 - [DNSKEY_RR_PROTO](#), 1749
 - [dnskey_t](#), 1750
 - [DNSSEC_DIGEST_SHA1](#), 1749
 - [DNSSEC_DIGEST_SHA256](#), 1749
 - [ds_rr_t](#), 1759
 - [ds_t](#), 1750
 - [IS_ROOT_LABEL](#), 1749
 - [NS_ALG_RSASHA1](#), 1749
 - [NS_ALG_RSASHA256](#), 1749
 - [NS_ALG_RSASHA512](#), 1749
 - [PUBLIC_FUNC_DECL](#), 1758, 1759
 - [ROOT_KEY_FILE](#), 1749
 - [rrsig_rr_t](#), 1759
 - [T_DNSKEY](#), 1750
 - [T_DS](#), 1750
 - [T_RRSIG](#), 1750
- [dnskey](#), 56
 - [algorithm](#), 56
 - [do_cache](#), 56
 - [dse](#), 56
 - [is_sep](#), 56
 - [is_zone](#), 57
 - [keytag](#), 57
 - [label](#), 57
 - [pubkey](#), 57
 - [rdata](#), 57
 - [rdlen](#), 57
 - [signkeys](#), 57
 - [validated](#), 57
- [DNSKEY_RR_FLAG_SEP](#)
 - [dns.h](#), 1748
- [DNSKEY_RR_FLAG_ZK](#)
 - [dns.h](#), 1748
- [dnskey_rr_flags_t](#)
 - [dns.h](#), 1759
- [DNSKEY_RR_LEN](#)
 - [dns.h](#), 1749
- [DNSKEY_RR_PROTO](#)
 - [dns.h](#), 1749
- [dnskey_t](#)
 - [dns.h](#), 1750
- [DNSSEC_DIGEST_SHA1](#)
 - [dns.h](#), 1749
- [DNSSEC_DIGEST_SHA256](#)
 - [dns.h](#), 1749
- [do_cache](#)
 - [dnskey](#), 56
- [do_ocsp_validation](#)
 - [ssl_pub.c](#), 1784
- [do_x509_hostname_check](#)
 - [ssl_pub.c](#), 1784
- [do_x509_validation](#)
 - [ssl_pub.c](#), 1784
- [doc_ctx](#)
 - [http_page_t](#), 77
- [doc_obj](#)
 - [http_page_t](#), 77
- [domain](#)
 - [dmtplib_session_t](#), 55
 - [magma_t](#), 93
 - [server_config_t](#), 127
 - [server_t](#), 131
 - [smtp_inbound_prefs_t](#), 142
 - [smtp_outbound_prefs_t](#), 149
- [domain_alloc](#)
 - [domains.c](#), 1244
 - [warehouse.h](#), 1253
- [domain_dkim](#)
 - [domains.c](#), 1245
 - [warehouse.h](#), 1253
- [domain_mailboxes](#)
 - [domains.c](#), 1245

- warehouse.h, 1254
- domain_restricted
 - domains.c, 1245
 - warehouse.h, 1254
- domain_spf
 - domains.c, 1245
 - warehouse.h, 1254
- domain_start
 - domains.c, 1246
 - warehouse.h, 1254
- domain_stop
 - domains.c, 1246
 - warehouse.h, 1255
- domain_t
 - warehouse.h, 1258
- domain_wildcard
 - domains.c, 1246
 - warehouse.h, 1255
- domains
 - domains.c, 1247
- domains.c
 - domain_alloc, 1244
 - domain_dkim, 1245
 - domain_mailboxes, 1245
 - domain_restricted, 1245
 - domain_spf, 1245
 - domain_start, 1246
 - domain_stop, 1246
 - domain_wildcard, 1246
 - domains, 1247
- DOMESTIC
 - checkers.h, 1262
- DONNA_INLINE
 - ed25519-donna-portable.h, 1656
- DONNA_NOINLINE
 - ed25519-donna-portable.h, 1656
- double_conv
 - numbers.c, 409
 - numbers.h, 425
- drec
 - dmtplib_session_t, 55
- ds, 59
 - algorithm, 59
 - digest, 59
 - digest_type, 59
 - diglen, 59
 - keytag, 59
 - label, 60
 - signkeys, 60
 - validated, 60
- ds_rr_t
 - dns.h, 1759
- ds_t
 - dns.h, 1750
- dse
 - dnskey, 56
- dspam.c
 - dspam_check, 1280
 - dspam_start, 1280
 - dspam_stop, 1280
 - dspam_train, 1280
 - lib_load_dspam, 1281
 - lib_version_dspam, 1281
 - sql_pool, 1281
- dspam_attach_d
 - symbols.c, 1989
 - symbols.h, 2029
- dspam_check
 - checkers.h, 1264
 - dspam.c, 1280
- dspam_create_d
 - symbols.c, 1989
 - symbols.h, 2030
- dspam_destroy_d
 - symbols.c, 1989
 - symbols.h, 2030
- dspam_detach_d
 - symbols.c, 1990
 - symbols.h, 2030
- dspam_init_driver_d
 - symbols.c, 1990
 - symbols.h, 2030
- dspam_process_d
 - symbols.c, 1990
 - symbols.h, 2030
- dspam_shutdown_driver_d
 - symbols.c, 1990
 - symbols.h, 2030
- dspam_start
 - checkers.h, 1264
 - dspam.c, 1280
- dspam_stop
 - checkers.h, 1265
 - dspam.c, 1280
- dspam_train
 - checkers.h, 1265
 - dspam.c, 1280
- dspam_version_d
 - symbols.c, 1990
 - symbols.h, 2030
- dtype
 - cached_object, 31
 - cached_store_t, 35
- dump
 - cached_store_t, 35
- dump_buf
 - misc_pub.c, 1557
- dump_buf_outer
 - misc_pub.c, 1557
- dump_error_stack
 - error.c, 1521
 - error.h, 1534
- dump_last_error
 - error.c, 1522

- error.h, 1534
- dx
 - dime_record_t, 38
 - dmtp_session_t, 55
- dx_connect_dual
 - dmtp_pub.c, 1727
- dx_connect_standard
 - dmtp_pub.c, 1727
- EC_ENCRYPT_CURVE
 - dcrypto.h, 1515
- EC_GROUP_clear_free_d
 - symbols.c, 1990
 - symbols.h, 2030
- EC_GROUP_free_d
 - symbols.c, 1990
 - symbols.h, 2030
- EC_GROUP_new_by_curve_name_d
 - symbols.c, 1990
 - symbols.h, 2030
- EC_GROUP_precompute_mult_d
 - symbols.c, 1990
 - symbols.h, 2030
- EC_GROUP_set_point_conversion_form_d
 - symbols.h, 2031
- EC_KEY_check_key_d
 - symbols.c, 1990
 - symbols.h, 2031
- EC_KEY_free_d
 - symbols.c, 1990
 - symbols.h, 2031
- EC_KEY_generate_key_d
 - symbols.c, 1991
 - symbols.h, 2031
- EC_KEY_get0_group_d
 - symbols.c, 1991
 - symbols.h, 2031
- EC_KEY_get0_private_key_d
 - symbols.c, 1991
 - symbols.h, 2031
- EC_KEY_get0_public_key_d
 - symbols.c, 1991
 - symbols.h, 2031
- EC_KEY_get_conv_form_d
 - symbols.h, 2031
- EC_KEY_new_by_curve_name_d
 - symbols.c, 1991
 - symbols.h, 2031
- EC_KEY_new_d
 - symbols.c, 1991
 - symbols.h, 2031
- EC_KEY_set_conv_form_d
 - symbols.h, 2032
- EC_KEY_set_group_d
 - symbols.c, 1991
 - symbols.h, 2032
- EC_KEY_set_private_key_d
 - symbols.c, 1991
 - symbols.h, 2032
- EC_KEY_set_public_key_d
 - symbols.h, 2032
- EC_POINT_cmp_d
 - symbols.h, 2032
- EC_POINT_free_d
 - symbols.c, 1991
 - symbols.h, 2032
- EC_POINT_hex2point_d
 - symbols.h, 2032
- EC_POINT_mul_d
 - symbols.h, 2032
- EC_POINT_new_d
 - symbols.c, 1991
 - symbols.h, 2032
- EC_POINT_oct2point_d
 - symbols.h, 2032
- EC_POINT_point2hex_d
 - symbols.h, 2032
- EC_POINT_point2oct_d
 - symbols.h, 2032
- EC_PUBKEY_SIZE
 - dcrypto.h, 1515
- ec_sign_data
 - crypto_pub.c, 1511
- ec_sign_sha_data
 - crypto_pub.c, 1511
- EC_SIGNING_CURVE
 - dcrypto.h, 1515
- ecdh1024
 - openssl.c, 1412
- ecdh512
 - openssl.c, 1412
- ECDH_compute_key_d
 - symbols.h, 2033
- ECDSA_do_sign_d
 - symbols.h, 2033
- ECDSA_do_verify_d
 - symbols.h, 2033
- ECDSA_SIG_free_d
 - symbols.c, 1992
 - symbols.h, 2033
- ECIES_PRIVATE_BINARY
 - cryptography/cryptography.h, 1349
- ECIES_PRIVATE_HEX
 - cryptography/cryptography.h, 1349
- ECIES_PUBLIC_BINARY
 - cryptography/cryptography.h, 1349
- ECIES_PUBLIC_HEX
 - cryptography/cryptography.h, 1349
- ECIES_CIPHER
 - cryptography/cryptography.h, 1348
- ecies_cipher_evpcryptography/ecies.c, 1394
- ecies_cipher_evcprecated/ecies.c, 1400
- ECIES_CURVE

- cryptography/cryptography.h, 1348
- ecies_curve_group
 - cryptography/ecies.c, 1394
 - deprecated/ecies.c, 1400
- ecies_decrypt
 - deprecated.h, 1499
 - deprecated/ecies.c, 1397
- ecies_encrypt
 - deprecated.h, 1499
 - deprecated/ecies.c, 1397
- ecies_env_derivation
 - crypto_pub.c, 1511
- ECIES_ENVELOPE
 - cryptography/cryptography.h, 1348
- ecies_envelope_derivation
 - deprecated.h, 1500
 - deprecated/ecies.c, 1397
- ecies_envelope_evp
 - cryptography/ecies.c, 1395
 - deprecated/ecies.c, 1400
 - encrypt_ctx, 62
- ecies_group
 - deprecated.h, 1500
 - deprecated/ecies.c, 1397
- ECIES_HMAC
 - cryptography/cryptography.h, 1348
- ecies_hmac_evp
 - cryptography/ecies.c, 1395
 - deprecated/ecies.c, 1400
- ecies_key_alloc
 - deprecated.h, 1500
 - deprecated/ecies.c, 1398
- ecies_key_create
 - deprecated.h, 1500
 - deprecated/ecies.c, 1398
- ecies_key_free
 - deprecated.h, 1500
 - deprecated/ecies.c, 1398
- ecies_key_private
 - deprecated.h, 1501
 - deprecated/ecies.c, 1398
- ecies_key_private_bin
 - deprecated.h, 1501
 - deprecated/ecies.c, 1398
- ecies_key_private_hex
 - deprecated.h, 1501
 - deprecated/ecies.c, 1399
- ecies_key_public
 - deprecated.h, 1501
 - deprecated/ecies.c, 1399
- ecies_key_public_bin
 - deprecated.h, 1502
 - deprecated/ecies.c, 1399
- ecies_key_public_hex
 - deprecated.h, 1502
 - deprecated/ecies.c, 1399
- ECIES_KEY_TYPE
 - cryptography/cryptography.h, 1349
- ecies_start
 - deprecated.h, 1502
 - deprecated/ecies.c, 1400
- ecies_stop
 - deprecated.h, 1502
 - deprecated/ecies.c, 1400
- ed25519-donna-batchverify.h
 - batch_heap, 1651
 - batch_point_buffer, 1652
 - ed25519_sign_open_batch, 1651
 - heap_batch_size, 1651
 - heap_index_t, 1651
 - max_batch_size, 1651
- ed25519-donna-impl-base.h
 - S1_SWINDOWSIZE, 1653
 - S1_TABLE_SIZE, 1653
 - S2_SWINDOWSIZE, 1653
 - S2_TABLE_SIZE, 1653
- ed25519-donna-impl-sse2.h
 - S1_SWINDOWSIZE, 1654
 - S1_TABLE_SIZE, 1654
 - S2_SWINDOWSIZE, 1654
 - S2_TABLE_SIZE, 1654
- ed25519-donna-portable-identify.h
 - OS_NIX, 1655
- ed25519-donna-portable.h
 - ALIGN, 1656
 - DONNA_INLINE, 1656
 - DONNA_NOINLINE, 1656
 - mul32x32_64, 1656
 - ROTL32, 1656
 - ROTR32, 1656
- ed25519-donna-sse2.c
 - ED25519_SSE2, 1675
 - ED25519_SUFFIX, 1675
- ed25519-donna.h
 - ED25519_32BIT, 1657
 - ge25519, 1657
 - ge25519_niels, 1657
 - ge25519_p1p1, 1657
 - ge25519_pniels, 1657
 - hash_512bits, 1657
- ed25519-hash.h
 - ed25519_hash_context, 1662
- ed25519-randombytes.h
 - ed25519_randombytes_unsafe, 1664
- ed25519-ref10.c
 - crypto_int32, 1677
 - crypto_int64, 1677
 - crypto_sign_open_ref10, 1678
 - crypto_sign_pk_ref10, 1678
 - crypto_sign_ref10, 1678
 - crypto_uint32, 1677
 - crypto_uint64, 1678
 - F, 1677
 - fe, 1678

- ed25519-ref10.h
 - crypto_sign_open_ref10, 1679
 - crypto_sign_pk_ref10, 1679
 - crypto_sign_ref10, 1679
- ed25519.h
 - curved25519_key, 1670
 - curved25519_scalarmult_basepoint_donna, 1670
 - ED25519_FN, 1669
 - ED25519_FN2, 1669
 - ED25519_FN3, 1669
 - ed25519_public_key, 1670
 - ed25519_publickey_donna, 1670
 - ed25519_randombytes_unsafe_donna, 1670
 - ed25519_secret_key, 1670
 - ed25519_sign_donna, 1670
 - ed25519_sign_open_batch_donna, 1670
 - ed25519_sign_open_donna, 1670
 - ed25519_signature, 1670
 - ED25519_SUFFIX, 1669
- ED25519_ERR
 - prime.h, 1904
- ED25519_PRIV
 - prime.h, 1904
- ED25519_PUB
 - prime.h, 1904
- ED25519_32BIT
 - ed25519-donna.h, 1657
- ED25519_64BIT_TABLES
 - curve25519-donna-64bit.h, 1639
- ed25519_alloc
 - prime/cryptography/cryptography.h, 1383
 - prime/cryptography/ed25519.c, 1666
- ED25519_FN
 - ed25519.h, 1669
- ED25519_FN2
 - ed25519.h, 1669
- ED25519_FN3
 - ed25519.h, 1669
- ed25519_free
 - prime/cryptography/cryptography.h, 1384
 - prime/cryptography/ed25519.c, 1666
- ed25519_generate
 - prime/cryptography/cryptography.h, 1384
 - prime/cryptography/ed25519.c, 1666
- ed25519_hash_context
 - ed25519-hash.h, 1662
- ED25519_KEY, 61
 - private_key, 61
 - public_key, 61
- ED25519_KEY_B64_SIZE
 - dcrypto.h, 1515
- ED25519_KEY_PRIV_LEN
 - prime.h, 1903
- ED25519_KEY_PUB_LEN
 - prime.h, 1903
- ED25519_KEY_SIZE
 - dcrypto.h, 1515
- ed25519_key_t
 - prime.h, 1910
- ed25519_key_type_t
 - prime.h, 1904
- ED25519_keypair_d
 - symbols.h, 2033
- ED25519_keypair_from_seed_d
 - symbols.h, 2033
- ed25519_private_get
 - prime/cryptography/cryptography.h, 1384
 - prime/cryptography/ed25519.c, 1666
- ed25519_private_set
 - prime/cryptography/cryptography.h, 1384
 - prime/cryptography/ed25519.c, 1667
- ed25519_public_get
 - prime/cryptography/cryptography.h, 1384
 - prime/cryptography/ed25519.c, 1667
- ed25519_public_key
 - ed25519.h, 1670
 - fuzz/ed25519-donna.h, 1658
- ed25519_public_set
 - prime/cryptography/cryptography.h, 1384
 - prime/cryptography/ed25519.c, 1667
- ed25519_publickey
 - dime/ed25519/ed25519.c, 1665
 - fuzz/ed25519-donna.h, 1659
- ed25519_publickey_donna
 - ed25519.h, 1670
- ed25519_publickey_sse2
 - fuzz/ed25519-donna.h, 1659
- ed25519_randombytes_unsafe
 - ed25519-randombytes.h, 1664
 - fuzz/ed25519-donna.h, 1659
- ed25519_randombytes_unsafe_donna
 - ed25519.h, 1670
- ed25519_randombytes_unsafe_sse2
 - fuzz/ed25519-donna.h, 1659
- ed25519_secret_key
 - ed25519.h, 1670
 - fuzz/ed25519-donna.h, 1658
- ED25519_SIG_B64_SIZE
 - dcrypto.h, 1516
- ED25519_SIG_SIZE
 - dcrypto.h, 1516
- ed25519_sign
 - dime/ed25519/ed25519.c, 1665
 - fuzz/ed25519-donna.h, 1659
 - prime/cryptography/cryptography.h, 1385
 - prime/cryptography/ed25519.c, 1667
- ED25519_sign_d
 - symbols.h, 2033
- ed25519_sign_data
 - crypto_pub.c, 1511
- ed25519_sign_donna
 - ed25519.h, 1670
- ed25519_sign_open
 - dime/ed25519/ed25519.c, 1665

- fuzz/ed25519-donna.h, 1659
- ed25519_sign_open_batch
 - ed25519-donna-batchverify.h, 1651
 - fuzz/ed25519-donna.h, 1659
- ed25519_sign_open_batch_donna
 - ed25519.h, 1670
- ed25519_sign_open_batch_sse2
 - fuzz/ed25519-donna.h, 1660
- ed25519_sign_open_donna
 - ed25519.h, 1670
- ed25519_sign_open_sse2
 - fuzz/ed25519-donna.h, 1660
- ed25519_sign_sse2
 - fuzz/ed25519-donna.h, 1660
- ed25519_signature
 - ed25519.h, 1670
 - fuzz/ed25519-donna.h, 1658
- ED25519_SIGNATURE_LEN
 - prime.h, 1903
- ED25519_SSE2
 - ed25519-donna-sse2.c, 1675
- ED25519_SUFFIX
 - ed25519-donna-sse2.c, 1675
 - ed25519.h, 1669
- ed25519_type
 - prime/cryptography/cryptography.h, 1385
 - prime/cryptography/ed25519.c, 1667
- ed25519_verify
 - prime/cryptography/cryptography.h, 1385
 - prime/cryptography/ed25519.c, 1667
- ED25519_verify_d
 - symbols.h, 2033
- ed25519_verify_sig
 - crypto_pub.c, 1512
- EMPTY
 - core.h, 229
- enable
 - magma_t, 93
- enable_core_dumps
 - magma_t, 94
- enabled
 - magma_t, 94
 - secure.c, 391
 - server_t, 131
- encode_rsa_pubkey
 - misc_pub.c, 1557
- encoding
 - mail_mime_t, 106
- encoding.c
 - libdime_base64_decode, 1825
 - libdime_base64_encode, 1825
- encoding.h
 - libdime_base64_decode, 1826
 - libdime_base64_encode, 1826
- encodings.h
 - BASE64_LINE_WRAP_CRLF, 239
 - BASE64_LINE_WRAP_LF, 239
 - BASE64_LINE_WRAP_NONE, 239
 - base64_decode, 239
 - base64_decode_mod, 239
 - base64_decode_opts, 240
 - base64_decoded_length, 240
 - base64_decoded_length_mod, 240
 - base64_encode, 240
 - base64_encode_mod, 241
 - base64_encode_opts, 241
 - base64_encode_wrap, 242
 - base64_encoded_length, 242
 - base64_encoded_length_mod, 242
 - base64_encoded_length_wrap, 242
 - BASE64_LINE_WRAP_LENGTH, 239
 - base64_wrap_t, 239
 - hex_count_st, 243
 - hex_decode_chr, 243
 - hex_decode_opts, 243
 - hex_decode_st, 243
 - hex_encode_chr, 244
 - hex_encode_opts, 244
 - hex_encode_st, 244
 - hex_encode_st_debug, 245
 - hex_valid_chr, 245
 - hex_valid_st, 245
 - mappings, 248
 - qp_decode, 246
 - qp_encode, 246
 - QP_LINE_WRAP_LENGTH, 239
 - url_decode, 246
 - url_encode, 247
 - URL_MAX_LENGTH, 239
 - url_valid_chr, 247
 - url_valid_st, 247
 - zbase32_decode, 247
 - zbase32_encode, 248
- encrypt.c
 - encrypt_ctx_free, 1828
 - encrypt_ctx_new, 1828
 - encrypt_deserialize_privkey, 1828
 - encrypt_deserialize_pubkey, 1828
 - encrypt_keypair_generate, 1828
 - LOG_OPENSSL_ERROR, 1827
- encrypt.h
 - encrypt_ctx_free, 1830
 - encrypt_ctx_new, 1830
 - encrypt_ctx_t, 1830
 - encrypt_deserialize_privkey, 1831
 - encrypt_deserialize_pubkey, 1831
 - encrypt_keypair_generate, 1831
- encrypt_aes_256
 - crypto_pub.c, 1512
- encrypt_ctx, 62
 - ecies_envelope_evp, 62
 - encryption_group, 62
- encrypt_ctx_free
 - encrypt.c, 1828

- encrypt.h, 1830
- encrypt_ctx_new
 - encrypt.c, 1828
 - encrypt.h, 1830
- encrypt_ctx_t
 - encrypt.h, 1830
- encrypt_deserialize_privkey
 - encrypt.c, 1828
 - encrypt.h, 1831
- encrypt_deserialize_pubkey
 - encrypt.c, 1828
 - encrypt.h, 1831
- encrypt_keypair_generate
 - encrypt.c, 1828
 - encrypt.h, 1831
- encrypt_keypair_t, 63
 - unused, 63
- encrypted
 - dmime_chunk_key_t, 41
- encrypted.c
 - encrypted_chunk_alloc, 1882
 - encrypted_chunk_buffer, 1882
 - encrypted_chunk_cleanup, 1882
 - encrypted_chunk_free, 1882
 - encrypted_chunk_get, 1882
 - encrypted_chunk_set, 1883
- encrypted_chunk_alloc
 - chunks.h, 1875
 - encrypted.c, 1882
- encrypted_chunk_buffer
 - chunks.h, 1875
 - encrypted.c, 1882
- encrypted_chunk_cleanup
 - chunks.h, 1875
 - encrypted.c, 1882
- encrypted_chunk_free
 - chunks.h, 1876
 - encrypted.c, 1882
- encrypted_chunk_get
 - chunks.h, 1876
 - encrypted.c, 1882
- encrypted_chunk_set
 - chunks.h, 1876
 - encrypted.c, 1883
- encrypted_message_alloc
 - providers/prime/messages/messages.c, 1088
 - providers/prime/messages/messages.h, 1186
- encrypted_message_cleanup
 - providers/prime/messages/messages.c, 1088
 - providers/prime/messages/messages.h, 1186
- encrypted_message_free
 - providers/prime/messages/messages.c, 1088
 - providers/prime/messages/messages.h, 1186
- encryption_group
 - encrypt_ctx, 62
- endpoint.c
 - portal_debug, 2181
- portal_endpoint, 2181
- portal_endpoint_ad, 2182
- portal_endpoint_alert_acknowledge, 2182
- portal_endpoint_alert_list, 2182
- portal_endpoint_aliases, 2183
- portal_endpoint_attachments_add, 2183
- portal_endpoint_attachments_progress, 2183
- portal_endpoint_attachments_remove, 2183
- portal_endpoint_auth, 2184
- portal_endpoint_compare, 2184
- portal_endpoint_config_edit, 2184
- portal_endpoint_config_load, 2185
- portal_endpoint_contacts_add, 2185
- portal_endpoint_contacts_copy, 2186
- portal_endpoint_contacts_edit, 2186
- portal_endpoint_contacts_list, 2186
- portal_endpoint_contacts_load, 2187
- portal_endpoint_contacts_move, 2187
- portal_endpoint_contacts_remove, 2187
- portal_endpoint_cookies, 2188
- portal_endpoint_error, 2188
- portal_endpoint_folders_add, 2188
- portal_endpoint_folders_list, 2189
- portal_endpoint_folders_remove, 2189
- portal_endpoint_folders_rename, 2190
- portal_endpoint_folders_tags, 2190
- portal_endpoint_logout, 2190
- portal_endpoint_messages_compose, 2191
- portal_endpoint_messages_copy, 2191
- portal_endpoint_messages_flag, 2191
- portal_endpoint_messages_list, 2191
- portal_endpoint_messages_load, 2192
- portal_endpoint_messages_move, 2192
- portal_endpoint_messages_remove, 2192
- portal_endpoint_messages_send, 2193
- portal_endpoint_messages_tag, 2193
- portal_endpoint_messages_tags, 2194
- portal_endpoint_response, 2194
- portal_endpoint_scrape, 2195
- portal_endpoint_scrape_add, 2195
- portal_endpoint_search, 2195
- portal_endpoint_sort, 2195
- portal_get_upload_attachment, 2196
- portal_meta, 2196
- portal_settings_changepass, 2196
- portal_settings_identity, 2197
- portal_upload, 2197
- portal_validate_request, 2197
- endpoints.c
 - api_endpoint_auth, 2172
 - api_endpoint_change_password, 2172
 - api_endpoint_delete_user, 2172
 - api_endpoint_register, 2172
- ends.c
 - st_cmp_ci_ends, 214
 - st_cmp_cs_ends, 214
- engine.c

- engine_compress, 1302
- engine_decompress, 1302
- engine/config/cache/cache.c
 - cache_alloc, 611
 - cache_config, 611
 - cache_free, 612
 - cache_output_help, 612
 - cache_output_settings, 612
 - cache_set_value, 613
 - cache_validate, 613
- engine/config/cache/cache.h
 - cache_alloc, 644
 - cache_config, 644
 - cache_free, 645
 - cache_output_help, 645
 - cache_output_settings, 645
 - cache_set_value, 646
 - cache_validate, 646
- engine/config/cache/keys.h
 - cache_keys, 663
- engine/config/config.h
 - CONFIG_CHECK_DIR_READABLE, 672
 - CONFIG_CHECK_DIR_READWRITE, 672
 - CONFIG_CHECK_EXISTS, 673
 - CONFIG_CHECK_FILE_READABLE, 673
 - CONFIG_CHECK_FILE_READWRITE, 673
 - CONFIG_CHECK_READWRITE, 673
 - MAGMA_BLACKLIST_INSTANCES, 673
 - MAGMA_CACHE_INSTANCES, 673
 - MAGMA_CACHE_SERVER_RETRY, 673
 - MAGMA_CACHE_SOCKET_TIMEOUT, 674
 - MAGMA_CONNECTION_BUFFER_SIZE, 674
 - MAGMA_CRYPTOGRAPHY_SEED_SIZE, 674
 - MAGMA_FILENAME_MAX, 674
 - MAGMA_FILEPATH_MAX, 674
 - MAGMA_HOSTNAME_MAX, 674
 - MAGMA_LOCATION_CACHE, 674
 - MAGMA_LOGS, 674
 - MAGMA_RELAY_INSTANCES, 674
 - MAGMA_RESOURCE_FONTS, 674
 - MAGMA_RESOURCE_LOCATION, 674
 - MAGMA_RESOURCE_PAGES, 675
 - MAGMA_RESOURCE_TEMPLATES, 675
 - MAGMA_RESOURCE_VIRUS, 675
 - MAGMA_SERVER_INSTANCES, 675
 - MAGMA_SMTP_LINE_WRAP_LENGTH, 675
 - MAGMA_SMTP_MAX_ADDRESS_SIZE, 675
 - MAGMA_SMTP_MAX_HELO_SIZE, 675
 - MAGMA_SMTP_MAX_MESSAGE_SIZE, 675
 - MAGMA_SMTP_RECIPIENT_LIMIT, 675
 - MAGMA_SMTP_RELAY_LIMIT, 675
 - MAGMA_THREAD_BUFFER_SIZE, 675
 - MAGMA_THREAD_STACK_SIZE, 676
 - MAGMA_WORKER_THREAD_LIMIT, 676
- engine/config/global/datatier.c
 - config_fetch_host_number, 682
 - config_fetch_settings, 682
- engine/config/global/keys.h
 - magma_keys, 664
- engine/config/relay/keys.h
 - relay_keys, 665
- engine/config/relay/relay.c
 - relay_alloc, 735
 - relay_config, 735
 - relay_counter, 736
 - relay_free, 736
 - relay_output_help, 736
 - relay_output_settings, 737
 - relay_set_value, 737
 - relay_validate, 737
- engine/config/servers/keys.h
 - server_keys, 666
- engine/config/servers/servers.h
 - DMTP, 753
 - GENERIC, 753
 - HTTP, 753
 - IMAP, 753
 - M_PORT, 753
 - M_PROTOCOL, 753
 - MOLTEN, 753
 - POP, 753
 - servers_alloc, 753
 - servers_config, 753
 - servers_encryption_start, 754
 - servers_encryption_stop, 754
 - servers_free, 754
 - servers_get_by_protocol, 755
 - servers_get_by_socket, 755
 - servers_get_count_using_port, 755
 - servers_network_start, 756
 - servers_network_stop, 756
 - servers_output_help, 756
 - servers_output_settings, 756
 - servers_set_value, 757
 - servers_validate, 757
 - SMTP, 753
 - SUBMISSION, 753
 - TCP_PORT, 753
 - TLS_PORT, 753
- engine/context/process.c
 - day, 317
 - exit_and_dump, 317
 - maint, 317
 - process_maint, 316
 - process_start, 316
 - process_stop, 317
- engine/context/thread.c
 - thread_start, 598
 - thread_stop, 598
- engine/status/statistics.c
 - count, 804
 - derived, 805
 - locks, 805
 - names, 805

- stats, 805
- stats_adjust_by_name, 799
- stats_adjust_by_num, 799
- stats_decrement_by_name, 800
- stats_decrement_by_num, 800
- stats_derived_count, 800
- stats_derived_name, 800
- stats_derived_value, 801
- stats_get_count, 801
- stats_get_name, 801
- stats_get_name_pos, 801
- stats_get_value_by_name, 802
- stats_get_value_by_num, 802
- stats_increment_by_name, 802
- stats_increment_by_num, 803
- stats_init, 803
- stats_set_by_name, 803
- stats_set_by_num, 803
- stats_shutdown, 804
- stats_sum_errors, 804
- values, 805
- ENGINE_cleanup_d
 - symbols.c, 1992
 - symbols.h, 2033
- engine_compress
 - compress.h, 1299
 - engine.c, 1302
- engine_decompress
 - compress.h, 1299
 - engine.c, 1302
- enqueue
 - controller.h, 777
 - queue.c, 784
- entire
 - mail_mime_t, 106
- entry_t
 - storage.h, 1964
- envelope
 - imap_fetch_dataitems_t, 78
- ephemeral
 - auth_stacie_t, 23
 - auth_t, 25
 - dmime_message_t, 47
- ephemeral.c
 - ephemeral_chunk_alloc, 1884
 - ephemeral_chunk_buffer, 1884
 - ephemeral_chunk_cleanup, 1884
 - ephemeral_chunk_free, 1884
 - ephemeral_chunk_get, 1884
 - ephemeral_chunk_set, 1885
- ephemeral_chunk_alloc
 - chunks.h, 1876
 - ephemeral.c, 1884
- ephemeral_chunk_buffer
 - chunks.h, 1876
 - ephemeral.c, 1884
- ephemeral_chunk_cleanup
 - chunks.h, 1876
 - ephemeral.c, 1884
- ephemeral_chunk_free
 - chunks.h, 1877
 - ephemeral.c, 1884
- ephemeral_chunk_get
 - chunks.h, 1877
 - ephemeral.c, 1884
- ephemeral_chunk_set
 - chunks.h, 1877
 - ephemeral.c, 1885
- equal.c
 - mm_cmp_ci_eq, 215
 - mm_cmp_cs_eq, 215
 - st_cmp_ci_eq, 216
 - st_cmp_cs_eq, 216
- ERR_BAD_PARAM
 - error.h, 1526
 - error_codes.c, 1690
 - error_codes.h, 1691
- ERR_clear_error_d
 - symbols.c, 1992
 - symbols.h, 2033
- ERR_CORRUPTION
 - error.h, 1526
- ERR_CRYPT
 - error_codes.c, 1690
 - error_codes.h, 1691
- err_desc_t, 64
 - errcode, 64
 - errmsg, 64
- ERR_ENCODING
 - error_codes.c, 1690
 - error_codes.h, 1691
- ERR_error_string_n_d
 - symbols.h, 2033
- ERR_FILE_IO
 - error_codes.c, 1690
 - error_codes.h, 1691
- ERR_free_strings_d
 - symbols.c, 1992
 - symbols.h, 2034
- ERR_get_error_d
 - symbols.c, 1992
 - symbols.h, 2034
- ERR_load_crypto_strings_d
 - symbols.c, 1992
 - symbols.h, 2034
- ERR_NOMEM
 - error.h, 1526
 - error_codes.c, 1690
 - error_codes.h, 1692
- ERR_OPENSSL
 - error.h, 1527
- ERR_peek_error_d
 - symbols.c, 1992
 - symbols.h, 2034

- ERR_peek_error_line_data_d
 - symbols.h, 2034
- ERR_PERM
 - error.h, 1527
- ERR_print_errors_fp_d
 - symbols.c, 1992
 - symbols.h, 2034
- ERR_put_error_d
 - symbols.h, 2034
- ERR_remove_thread_state_d
 - symbols.c, 1992
 - symbols.h, 2034
- ERR_RESOLVER
 - error.h, 1527
- ERR_STACK_SIZE
 - error.h, 1527
- ERR_SYSCALL
 - error.h, 1527
- ERR_UNSPEC
 - error.h, 1527
- errcode
 - err_desc_t, 64
- ERRCODE_BAD_PARAM
 - error_codes.h, 1691
- ERRCODE_CRYPT0
 - error_codes.h, 1691
- ERRCODE_ENCODING
 - error_codes.h, 1691
- ERRCODE_FILE_IO
 - error_codes.h, 1691
- ERRCODE_NOMEM
 - error_codes.h, 1691
- errinfo_t
 - error.h, 1533
- errmsg
 - err_desc_t, 64
- errno_name
 - core/host/errors.c, 269
 - host.h, 292
- error.c
 - __attribute__, 1520
 - _clear_error_stack, 1520
 - _create_new_error, 1520
 - _dump_error, 1520
 - _push_error_stack, 1521
 - _push_error_stack_fmt, 1521
 - _push_error_stack_openssl, 1521
 - _push_error_stack_resolver, 1521
 - _push_error_stack_syscall, 1521
 - _t_err_stack, 1523
 - dump_error_stack, 1521
 - dump_last_error, 1522
 - get_error_string, 1522
 - get_first_error, 1522
 - get_last_error, 1522
 - get_last_error_code, 1522
 - pop_last_error, 1523
- error.h
 - __attribute__, 1533
 - _clear_error_stack, 1533
 - _create_new_error, 1533
 - _dump_error, 1533
 - _push_error_stack, 1533
 - _push_error_stack_fmt, 1533
 - _push_error_stack_openssl, 1534
 - _push_error_stack_resolver, 1534
 - _push_error_stack_syscall, 1534
 - dump_error_stack, 1534
 - dump_last_error, 1534
 - ERR_BAD_PARAM, 1526
 - ERR_CORRUPTION, 1526
 - ERR_NOMEM, 1526
 - ERR_OPENSSL, 1527
 - ERR_PERM, 1527
 - ERR_RESOLVER, 1527
 - ERR_STACK_SIZE, 1527
 - ERR_SYSCALL, 1527
 - ERR_UNSPEC, 1527
 - errinfo_t, 1533
 - get_error_string, 1534
 - get_first_error, 1535
 - get_last_error, 1535
 - get_last_error_code, 1535
 - pop_last_error, 1535
 - PUBLIC_FUNC_DECL, 1528
 - PUBLIC_FUNC_DECL_VA, 1528
 - PUBLIC_FUNC_IMPL, 1528
 - PUBLIC_FUNC_IMPL_VA1, 1529
 - PUBLIC_FUNC_IMPL_VA1_RET, 1529
 - PUBLIC_FUNC_IMPL_VA2, 1529
 - PUBLIC_FUNC_IMPL_VA2_RET, 1529
 - PUBLIC_FUNC_IMPL_VOID, 1529
 - PUBLIC_FUNC_PROLOGUE, 1529
 - PUBLIC_FUNCTION_IMPLEMENT, 1529
 - PUBLIC_FUNCTION_IMPLEMENT_VOID, 1530
 - PUSH_ERROR, 1530
 - PUSH_ERROR_FMT, 1530
 - PUSH_ERROR_OPENSSL, 1530
 - PUSH_ERROR_RESOLVER, 1530
 - PUSH_ERROR_SYSCALL, 1531
 - RET_ERROR_CUST, 1531
 - RET_ERROR_CUST_FMT, 1531
 - RET_ERROR_INT, 1531
 - RET_ERROR_INT_FMT, 1531
 - RET_ERROR_PTR, 1532
 - RET_ERROR_PTR_FMT, 1532
 - RET_ERROR_UINT, 1532
 - RET_ERROR_UINT_FMT, 1532
- error_codes.h
 - ERRCODE_BAD_PARAM, 1691
 - ERRCODE_CRYPT0, 1691
 - ERRCODE_ENCODING, 1691
 - ERRCODE_FILE_IO, 1691
 - ERRCODE_NOMEM, 1691

error_codes.c
 ERR_BAD_PARAM, 1690
 ERR_CRYPT, 1690
 ERR_ENCODING, 1690
 ERR_FILE_IO, 1690
 ERR_NOMEM, 1690
 error_codes.h
 dime_errcode_t, 1691
 ERR_BAD_PARAM, 1691
 ERR_CRYPT, 1691
 ERR_ENCODING, 1691
 ERR_FILE_IO, 1691
 ERR_NOMEM, 1692
 esmtp
 smtp_session_t, 152
 EVP_aes_256_cbc_d
 symbols.c, 1992
 symbols.h, 2034
 EVP_aes_256_gcm_d
 symbols.c, 1992
 symbols.h, 2034
 EVP_CIPHER_block_size_d
 symbols.c, 1992
 symbols.h, 2034
 EVP_CIPHER_CTX_block_size_d
 symbols.c, 1993
 symbols.h, 2034
 EVP_CIPHER_CTX_cleanup_d
 symbols.c, 1993
 symbols.h, 2034
 EVP_CIPHER_CTX_ctrl_d
 symbols.h, 2035
 EVP_CIPHER_CTX_flags_d
 symbols.h, 2035
 EVP_CIPHER_CTX_free_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_CTX_init_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_CTX_iv_length_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_CTX_key_length_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_CTX_new_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_CTX_set_padding_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_flags_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_iv_length_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_key_length_d
 symbols.c, 1993
 symbols.h, 2035
 EVP_CIPHER_nid_d
 symbols.c, 1994
 symbols.h, 2036
 EVP_cleanup_d
 symbols.c, 1994
 symbols.h, 2036
 EVP_DecryptFinal_ex_d
 symbols.h, 2036
 EVP_DecryptInit_ex_d
 symbols.h, 2036
 EVP_DecryptUpdate_d
 symbols.h, 2036
 EVP_Digest_d
 symbols.h, 2036
 EVP_DigestFinal_d
 symbols.h, 2036
 EVP_DigestFinal_ex_d
 symbols.h, 2036
 EVP_DigestInit_d
 symbols.c, 1994
 symbols.h, 2036
 EVP_DigestInit_ex_d
 symbols.h, 2036
 EVP_DigestUpdate_d
 symbols.h, 2036
 EVP_EncryptFinal_ex_d
 symbols.h, 2036
 EVP_EncryptInit_ex_d
 symbols.h, 2037
 EVP_EncryptUpdate_d
 symbols.h, 2037
 EVP_get_cipherbyname_d
 symbols.c, 1994
 symbols.h, 2037
 EVP_get_digestbyname_d
 symbols.c, 1994
 symbols.h, 2037
 EVP_md4_d
 symbols.c, 1994
 symbols.h, 2037
 EVP_md5_d
 symbols.c, 1994
 symbols.h, 2037
 EVP_MD_CTX_cleanup_d
 symbols.c, 1994
 symbols.h, 2037
 EVP_MD_CTX_init_d
 symbols.c, 1994
 symbols.h, 2037
 EVP_MD_size_d
 symbols.c, 1994
 symbols.h, 2037
 EVP_MD_type_d

- symbols.c, [1994](#)
 - symbols.h, [2037](#)
- EVP_PKEY_new_d
 - symbols.c, [1994](#)
 - symbols.h, [2037](#)
- EVP_PKEY_set1_RSA_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- EVP_ripemd160_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- EVP_sha1_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- EVP_sha224_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- EVP_sha256_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- EVP_sha384_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- EVP_sha512_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- EVP_sha_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- EVP_VerifyFinal_d
 - symbols.h, [2038](#)
- exit_and_dump
 - args.c, [761](#)
 - engine/context/process.c, [317](#)
 - global.c, [727](#)
- expiration
 - cached_object, [32](#)
- expiry
 - dime_record_t, [38](#)
- expression
 - smtp_inbound_filter_t, [139](#)
- expunge
 - __attribute__, [18](#)
- extension
 - media_type_t, [109](#)
- F
 - curve25519-donna-32bit.h, [1637](#), [1638](#)
 - curve25519-donna-64bit.h, [1639](#)
 - curve25519-donna-sse2.h, [1643](#)
 - ed25519-ref10.c, [1677](#)
- family
 - ip_t, [83](#)
- fe
 - curve25519-ref10.c, [1671](#)
 - ed25519-ref10.c, [1678](#)
- fe_0
 - curve25519-ref10.c, [1672](#)
- fe_1
 - curve25519-ref10.c, [1672](#)
- fe_add
 - curve25519-ref10.c, [1672](#)
- fe_copy
 - curve25519-ref10.c, [1672](#)
- fe_cswap
 - curve25519-ref10.c, [1672](#)
- fe_frombytes
 - curve25519-ref10.c, [1672](#)
- fe_invert
 - curve25519-ref10.c, [1672](#)
- fe_mul
 - curve25519-ref10.c, [1673](#)
- fe_mul121666
 - curve25519-ref10.c, [1673](#)
- fe_sq
 - curve25519-ref10.c, [1673](#)
- fe_sub
 - curve25519-ref10.c, [1673](#)
- fe_tobytes
 - curve25519-ref10.c, [1673](#)
- fetch.c
 - imap_duplicate_messages, [2115](#)
 - imap_fetch_body, [2115](#)
 - imap_fetch_body_header, [2116](#)
 - imap_fetch_body_mime, [2116](#)
 - imap_fetch_body_part, [2116](#)
 - imap_fetch_body_portion, [2116](#)
 - imap_fetch_body_tag, [2116](#)
 - imap_fetch_bodystructure, [2116](#)
 - imap_fetch_envelope, [2117](#)
 - imap_fetch_free_items, [2117](#)
 - imap_fetch_message, [2117](#)
 - imap_fetch_parse_partial, [2117](#)
 - imap_fetch_return_header, [2117](#)
 - imap_fetch_return_message, [2118](#)
 - imap_fetch_return_mime, [2118](#)
 - imap_fetch_return_text, [2118](#)
 - imap_narrow_messages, [2118](#)
 - imap_parse_dataitems, [2118](#)
 - imap_valid_sequence, [2118](#)
- fetch_response.c
 - imap_fetch_response_add, [2120](#)
 - imap_fetch_response_free, [2120](#)
- FETCH_SIGNATURE
 - queries.h, [2072](#)
- field
 - smtp_inbound_filter_t, [139](#)
- field_data_t
 - signet/common.h, [1576](#)
- FIELD_NAME_MAX_SIZE
 - signet/common.h, [1574](#)
- fields
 - signet_t, [138](#)
- fields.c

- prime_field_get, [1932](#)
- prime_field_size_length, [1932](#)
- prime_field_size_max, [1932](#)
- prime_field_write, [1932](#)
- file
 - magma_keys_t, [86](#)
 - magma_t, [94](#)
- file_accessible
 - files.c, [274](#)
 - host.h, [292](#)
- file_load
 - files.c, [274](#)
 - host.h, [292](#)
- file_read
 - files.c, [274](#)
 - host.h, [293](#)
- file_readwritable
 - files.c, [275](#)
 - host.h, [293](#)
- file_temp_handle
 - files.c, [275](#)
 - host.h, [293](#)
- file_world_accessible
 - files.c, [275](#)
 - host.h, [294](#)
- files.c
 - file_accessible, [274](#)
 - file_load, [274](#)
 - file_read, [274](#)
 - file_readwritable, [275](#)
 - file_temp_handle, [275](#)
 - file_world_accessible, [275](#)
- filters
 - smtp_inbound_prefs_t, [142](#)
- find_cached_object
 - cache_pub.c, [1709](#)
- find_cached_object_cmp
 - cache_pub.c, [1709](#)
- first
 - imap_folder_status_t, [82](#)
- fl
 - multi_t, [111](#)
- flags
 - imap_fetch_dataitems_t, [78](#)
 - object_chunk, [117](#)
- fletcher.c
 - hash_fletcher32, [199](#)
- float_conv
 - numbers.c, [409](#)
 - numbers.h, [425](#)
- FMMESSAGE_MAGIC_1
 - objects/messages/messages.h, [1176](#)
- FMMESSAGE_MAGIC_2
 - objects/messages/messages.h, [1176](#)
- FMMESSAGE_OPT_COMPRESSED
 - objects/messages/messages.h, [1176](#)
- FMMESSAGE_OPT_ENCRYPTED
 - objects/messages/messages.h, [1177](#)
- folder.c
 - folder_count, [277](#)
 - folder_exists, [277](#)
- folder_count
 - folder.c, [277](#)
 - host.h, [294](#)
- folder_exists
 - folder.c, [277](#)
 - host.h, [294](#)
- FOLDER_LENGTH_LIMIT
 - folders.h, [1074](#)
- FOLDER_RECURSION_LIMIT
 - folders.h, [1074](#)
- foldernum
 - imap_folder_status_t, [82](#)
 - smtp_inbound_filter_t, [139](#)
 - smtp_inbound_prefs_t, [142](#)
- folders.h
 - __attribute__, [1075](#)
 - contact_folder_alloc, [1075](#)
 - contact_folder_create, [1075](#)
 - contact_folder_free, [1076](#)
 - contact_folder_remove, [1076](#)
 - contact_folder_rename, [1076](#)
 - contact_folder_t, [1074](#)
 - FOLDER_LENGTH_LIMIT, [1074](#)
 - FOLDER_RECURSION_LIMIT, [1074](#)
 - M_FOLDER_CONTACTS, [1075](#)
 - M_FOLDER_MESSAGES, [1075](#)
 - magma_folder_alloc, [1077](#)
 - magma_folder_children, [1077](#)
 - magma_folder_delete, [1078](#)
 - magma_folder_fetch, [1078](#)
 - magma_folder_find_full_name, [1078](#)
 - magma_folder_find_name, [1079](#)
 - magma_folder_find_number, [1079](#)
 - magma_folder_free, [1079](#)
 - magma_folder_funcs, [1080](#)
 - magma_folder_insert, [1080](#)
 - magma_folder_name, [1080](#)
 - magma_folder_rename, [1081](#)
 - magma_folder_t, [1082](#)
 - message_folder_alloc, [1081](#)
 - message_folder_create, [1081](#)
 - message_folder_free, [1082](#)
 - message_folder_remove, [1082](#)
 - message_folder_t, [1074](#)
- folders/find.c
 - magma_folder_find_full_name, [1057](#)
 - magma_folder_find_name, [1057](#)
 - magma_folder_find_number, [1058](#)
- fonts
 - content.c, [2109](#)
 - magma_t, [94](#)
- FOREIGN
 - checkers.h, [1262](#)

- FOREIGNDATA
 - strings.h, [543](#)
- formats.h
 - nvp_alloc, [394](#)
 - nvp_free, [394](#)
 - nvp_init, [394](#)
 - nvp_parse, [395](#)
- forwarded
 - smtp_inbound_prefs_t, [143](#)
- fp_author
 - dmime_object_t, [49](#)
- fp_destination
 - dmime_object_t, [49](#)
- fp_origin
 - dmime_object_t, [49](#)
- fp_recipient
 - dmime_object_t, [49](#)
- free_ec_key
 - crypto_pub.c, [1512](#)
- free_ed25519_key
 - crypto_pub.c, [1512](#)
- free_ed25519_key_chain
 - crypto_pub.c, [1512](#)
- freetype.c
 - lib_load_freetype, [1832](#)
 - lib_magma, [1832](#)
 - lib_version_freetype, [1832](#)
- from
 - mail_message_t, [104](#)
 - smtp_message_t, [147](#)
- FT_Done_FreeType_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- FT_Init_FreeType_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- FT_Library_Version_d
 - symbols.c, [1995](#)
 - symbols.h, [2038](#)
- function
 - magma_t, [94](#)
 - queue_t, [121](#)
- fuzz-curve25519.c
 - main, [1680](#)
 - prng, [1680](#)
 - quarter, [1680](#)
 - rotl32, [1680](#)
- fuzz-ed25519.c
 - generated_data, [1682](#)
 - main, [1682](#)
 - prng, [1682](#)
 - quarter, [1681](#)
 - random_data, [1682](#)
 - rotl32, [1681](#)
- fuzz/ed25519-donna.h
 - curved25519_key, [1658](#)
 - curved25519_scalarmult_basepoint, [1659](#)
 - curved25519_scalarmult_basepoint_sse2, [1659](#)
 - ed25519_public_key, [1658](#)
 - ed25519_publickey, [1659](#)
 - ed25519_publickey_sse2, [1659](#)
 - ed25519_randombytes_unsafe, [1659](#)
 - ed25519_randombytes_unsafe_sse2, [1659](#)
 - ed25519_secret_key, [1658](#)
 - ed25519_sign, [1659](#)
 - ed25519_sign_open, [1659](#)
 - ed25519_sign_open_batch, [1659](#)
 - ed25519_sign_open_batch_sse2, [1660](#)
 - ed25519_sign_open_sse2, [1660](#)
 - ed25519_sign_sse2, [1660](#)
 - ed25519_signature, [1658](#)
- gd.c
 - lib_load_gd, [1833](#)
 - lib_version_gd, [1833](#)
- gdFree_d
 - symbols.c, [1995](#)
 - symbols.h, [2039](#)
- gdImageColorResolve_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- gdImageCreate_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- gdImageDestroy_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- gdImageGifPtr_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- gdImageJpegPtr_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- gdImageSetPixel_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- gdImageStringFT_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- gdVersionString_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- ge25519
 - ed25519-donna.h, [1657](#)
- ge25519_niels
 - ed25519-donna.h, [1657](#)
- ge25519_niels_t, [65](#)
 - t2d, [65](#)
 - xaddy, [65](#)
 - ysubx, [65](#)
- ge25519_p1p1
 - ed25519-donna.h, [1657](#)
- ge25519_p1p1_t, [66](#)
 - t, [66](#)

- x, 66
- y, 66
- z, 66
- ge25519_pniels
 - ed25519-donna.h, 1657
- ge25519_pniels_t, 67
 - t2d, 67
 - xaddy, 67
 - ysubx, 67
 - z, 67
- ge25519_t, 68
 - t, 68
 - x, 68
 - y, 68
 - z, 68
- ge_cached, 69
 - T2d, 69
 - YminusX, 69
 - YplusX, 69
 - Z, 69
- ge_p1p1, 70
 - T, 70
 - X, 70
 - Y, 70
 - Z, 70
- ge_p2, 71
 - X, 71
 - Y, 71
 - Z, 71
- ge_p3, 72
 - T, 72
 - X, 72
 - Y, 72
 - Z, 72
- ge_precomp, 73
 - xy2d, 73
 - yminusx, 73
 - yplusx, 73
- general.c
 - dime_number_to_str, 1786
 - signet_org_field_keys, 1787
 - signet_ssr_field_keys, 1787
 - signet_user_field_keys, 1787
 - SKEY_EMPTY, 1786
 - SKEY_SIZE1, 1786
 - SKEY_SIZE2, 1786
- generate_ec_keypair
 - crypto_pub.c, 1512
- generate_ed25519_keypair
 - crypto_pub.c, 1512
- generated_data
 - fuzz-ed25519.c, 1682
- generated_data_t, 74
 - pk, 74
 - sig, 74
 - valid, 74
- GENERIC
 - engine/config/servers/servers.h, 753
- get_cache_location
 - cache_pub.c, 1709
- get_cache_obj_data
 - cache_pub.c, 1709
- get_cert_subject_cn
 - ssl_pub.c, 1784
- get_chr_date
 - misc_pub.c, 1558
- get_dbg_level
 - misc_pub.c, 1558
- get_dime_dir_location
 - cache_pub.c, 1709
- get_dime_record
 - mrec_pub.c, 1768
- get_dime_record_from_file
 - mrec_pub.c, 1768
- get_error_string
 - error.c, 1522
 - error.h, 1534
- get_first_error
 - error.c, 1522
 - error.h, 1535
- get_header_opt
 - servers/http/http.h, 856
 - servers/http/parse.c, 2084
- get_header_value_noopt
 - servers/http/http.h, 856
 - servers/http/parse.c, 2084
- get_last_error
 - error.c, 1522
 - error.h, 1535
- get_last_error_code
 - error.c, 1522
 - error.h, 1535
- get_random_bytes
 - crypto_pub.c, 1512
- get_signet
 - dmtip_pub.c, 1727
- get_x509_cert_sha_hash
 - misc_pub.c, 1558
- global.c
 - cmdline_config_data, 727
 - config_free, 723
 - config_key_lookup, 723
 - config_load_cmdline_settings, 723
 - config_load_database_settings, 724
 - config_load_defaults, 724
 - config_load_file_settings, 724
 - config_output_help, 725
 - config_output_settings, 725
 - config_output_value, 725
 - config_output_value_generic, 725
 - config_validate_settings, 726
 - config_value_set, 726
 - exit_and_dump, 727
 - magma, 727

- threadBuffer, 728
- global.h
 - config_fetch_host_number, 730
 - config_fetch_settings, 730
 - config_free, 730
 - config_key_lookup, 730
 - config_load_cmdline_settings, 731
 - config_load_database_settings, 731
 - config_load_defaults, 732
 - config_load_file_settings, 732
 - config_output_help, 732
 - config_output_settings, 732
 - config_output_value, 733
 - config_output_value_generic, 733
 - config_validate_settings, 733
 - config_value_set, 734
- greylist
 - smtp_inbound_prefs_t, 143
- greytime
 - smtp_inbound_prefs_t, 143
- GZIP_COMPRESSION_ENABLED
 - dmessage/common.h, 1570
- hash.c
 - hash_digest, 1402
 - hash_md4, 1402
 - hash_md5, 1402
 - hash_ripemd160, 1402
 - hash_sha, 1402
 - hash_sha1, 1403
 - hash_sha224, 1403
 - hash_sha256, 1403
 - hash_sha384, 1403
 - hash_sha512, 1403
- hash_512bits
 - ed25519-donna.h, 1657
- hash_adler32
 - adler.c, 189
 - checksum.h, 192
- hash_digest
 - cryptography/cryptography.h, 1367
 - hash.c, 1402
- hash_fletcher32
 - checksum.h, 192
 - fletcher.c, 199
- hash_md4
 - cryptography/cryptography.h, 1367
 - hash.c, 1402
- hash_md5
 - cryptography/cryptography.h, 1367
 - hash.c, 1402
- hash_murmur32
 - checksum.h, 193
 - murmur.c, 200
- hash_murmur64
 - checksum.h, 193
 - murmur.c, 200
- hash_ripemd160
 - cryptography/cryptography.h, 1367
 - hash.c, 1402
- hash_sha
 - cryptography/cryptography.h, 1367
 - hash.c, 1402
- hash_sha1
 - cryptography/cryptography.h, 1367
 - hash.c, 1403
- hash_sha224
 - cryptography/cryptography.h, 1367
 - hash.c, 1403
- hash_sha256
 - cryptography/cryptography.h, 1368
 - hash.c, 1403
- hash_sha384
 - cryptography/cryptography.h, 1368
 - hash.c, 1403
- hash_sha512
 - cryptography/cryptography.h, 1368
 - hash.c, 1403
- hashed.c
 - __attribute__, 331
 - hashed_alloc, 331
 - hashed_bucket, 331
 - hashed_bucket_add, 331
 - hashed_bucket_alloc, 331
 - hashed_bucket_find_key, 332
 - hashed_bucket_get_ptr, 332
 - hashed_bucket_set_ptr, 332
 - hashed_bucket_t, 334
 - hashed_cursor_active, 332
 - hashed_cursor_alloc, 332
 - hashed_cursor_free, 332
 - hashed_cursor_key_active, 333
 - hashed_cursor_key_next, 333
 - hashed_cursor_next, 333
 - hashed_cursor_reset, 333
 - hashed_cursor_t, 334
 - hashed_cursor_value_active, 333
 - hashed_cursor_value_next, 333
 - hashed_delete, 333
 - hashed_find, 334
 - hashed_free, 334
 - hashed_index_t, 334
 - hashed_insert, 334
 - hashed_truncate, 334
 - MAGMA_HASHED_BUCKETS, 330
- hashed_alloc
 - hashed.c, 331
 - indexes.h, 338
- hashed_bucket
 - hashed.c, 331
- hashed_bucket_add
 - hashed.c, 331
- hashed_bucket_alloc
 - hashed.c, 331

- hashed_bucket_find_key
 - hashed.c, [332](#)
- hashed_bucket_get_ptr
 - hashed.c, [332](#)
- hashed_bucket_set_ptr
 - hashed.c, [332](#)
- hashed_bucket_t
 - hashed.c, [334](#)
- hashed_cursor_active
 - hashed.c, [332](#)
- hashed_cursor_alloc
 - hashed.c, [332](#)
- hashed_cursor_free
 - hashed.c, [332](#)
- hashed_cursor_key_active
 - hashed.c, [333](#)
- hashed_cursor_key_next
 - hashed.c, [333](#)
- hashed_cursor_next
 - hashed.c, [333](#)
- hashed_cursor_reset
 - hashed.c, [333](#)
- hashed_cursor_t
 - hashed.c, [334](#)
- hashed_cursor_value_active
 - hashed.c, [333](#)
- hashed_cursor_value_next
 - hashed.c, [333](#)
- hashed_delete
 - hashed.c, [333](#)
- hashed_find
 - hashed.c, [334](#)
- hashed_free
 - hashed.c, [334](#)
- hashed_index_t
 - hashed.c, [334](#)
- hashed_insert
 - hashed.c, [334](#)
- hashed_truncate
 - hashed.c, [334](#)
- head
 - cached_store_t, [35](#)
- header
 - mail_mime_t, [106](#)
- HEADER_TYPE_CC
 - parse.h, [1599](#)
- HEADER_TYPE_DATE
 - parse.h, [1599](#)
- HEADER_TYPE_FROM
 - parse.h, [1600](#)
- HEADER_TYPE_NONE
 - parse.h, [1600](#)
- HEADER_TYPE_ORGANIZATION
 - parse.h, [1600](#)
- HEADER_TYPE_SUBJECT
 - parse.h, [1600](#)
- HEADER_TYPE_TO
 - parse.h, [1599](#)
- header_length
 - mail_message_t, [104](#)
 - smtp_message_t, [147](#)
- headers
 - __attribute__, [18](#)
 - dmime_common_headers_t, [43](#)
- HEAP
 - strings.h, [543](#)
- heap
 - batch_heap_t, [28](#)
- heap_batch_size
 - ed25519-donna-batchverify.h, [1651](#)
- heap_index_t
 - ed25519-donna-batchverify.h, [1651](#)
- helo
 - smtp_session_t, [152](#)
- helo_length_limit
 - magma_t, [94](#)
- helpers.c
 - api_error, [2173](#)
 - api_response, [2173](#)
 - jansson_flags, [2174](#)
- HEX
 - signet/common.h, [1576](#)
- hex.c
 - hex_count_st, [249](#)
 - hex_decode_chr, [249](#)
 - hex_decode_opts, [250](#)
 - hex_decode_st, [250](#)
 - hex_encode_chr, [250](#)
 - hex_encode_opts, [251](#)
 - hex_encode_st, [251](#)
 - hex_encode_st_debug, [251](#)
 - hex_valid_chr, [252](#)
 - hex_valid_st, [252](#)
- hex_count_st
 - encodings.h, [243](#)
- hex.c, [249](#)
- hex_decode_chr
 - encodings.h, [243](#)
- hex.c, [249](#)
- hex_decode_opts
 - encodings.h, [243](#)
- hex.c, [250](#)
- hex_decode_st
 - encodings.h, [243](#)
- hex.c, [250](#)
- hex_digit_to_int
 - sds.c, [1694](#)
- hex_encode
 - misc_pub.c, [1558](#)
- hex_encode_chr
 - encodings.h, [244](#)
- hex.c, [250](#)
- hex_encode_opts
 - encodings.h, [244](#)

- hex.c, [251](#)
- hex_encode_st
 - encodings.h, [244](#)
 - hex.c, [251](#)
- hex_encode_st_debug
 - encodings.h, [245](#)
 - hex.c, [251](#)
- hex_valid_chr
 - encodings.h, [245](#)
 - hex.c, [252](#)
- hex_valid_st
 - encodings.h, [245](#)
 - hex.c, [252](#)
- HMAC_CTX_cleanup_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- HMAC_CTX_init_d
 - symbols.c, [1996](#)
 - symbols.h, [2039](#)
- hmac_digest
 - deprecated.h, [1503](#)
 - deprecated/hmac.c, [1406](#)
- HMAC_Final_d
 - symbols.h, [2039](#)
- HMAC_Init_ex_d
 - symbols.h, [2040](#)
- hmac_md4
 - deprecated.h, [1503](#)
 - deprecated/hmac.c, [1406](#)
- hmac_md5
 - deprecated.h, [1503](#)
 - deprecated/hmac.c, [1406](#)
- hmac_ripemd160
 - deprecated.h, [1503](#)
 - deprecated/hmac.c, [1407](#)
- hmac_sha
 - deprecated.h, [1503](#)
 - deprecated/hmac.c, [1407](#)
- hmac_sha1
 - deprecated.h, [1503](#)
 - deprecated/hmac.c, [1407](#)
- hmac_sha224
 - deprecated.h, [1503](#)
 - deprecated/hmac.c, [1407](#)
- hmac_sha256
 - deprecated.h, [1504](#)
 - deprecated/hmac.c, [1407](#)
- hmac_sha384
 - deprecated.h, [1504](#)
 - deprecated/hmac.c, [1407](#)
- hmac_sha512
 - deprecated.h, [1504](#)
 - deprecated/hmac.c, [1407](#)
- HMAC_Update_d
 - symbols.h, [2040](#)
- host
 - __attribute__, [18](#)
 - magma_t, [94](#), [95](#)
- host.c
 - host_platform, [278](#)
 - host_version, [278](#)
- host.h
 - backtrace_print, [288](#)
 - color_blue, [288](#)
 - color_blue_bold, [288](#)
 - color_blue_intense, [288](#)
 - color_blue_intense_bold, [288](#)
 - color_blue_underline, [288](#)
 - color_cyan, [288](#)
 - color_cyan_bold, [288](#)
 - color_cyan_intense, [289](#)
 - color_cyan_intense_bold, [289](#)
 - color_cyan_underline, [289](#)
 - color_green, [289](#)
 - color_green_bold, [289](#)
 - color_green_intense, [289](#)
 - color_green_intense_bold, [289](#)
 - color_green_underline, [289](#)
 - color_purple, [289](#)
 - color_purple_bold, [290](#)
 - color_purple_intense, [290](#)
 - color_purple_intense_bold, [290](#)
 - color_purple_underline, [290](#)
 - color_red, [290](#)
 - color_red_bold, [290](#)
 - color_red_intense, [290](#)
 - color_red_intense_bold, [290](#)
 - color_red_underline, [290](#)
 - color_reset, [291](#)
 - color_supported, [291](#)
 - color_white, [291](#)
 - color_white_bold, [291](#)
 - color_white_intense, [291](#)
 - color_white_intense_bold, [291](#)
 - color_white_underline, [291](#)
 - color_yellow, [291](#)
 - color_yellow_bold, [292](#)
 - color_yellow_intense, [292](#)
 - color_yellow_intense_bold, [292](#)
 - color_yellow_underline, [292](#)
 - errno_name, [292](#)
 - file_accessible, [292](#)
 - file_load, [292](#)
 - file_read, [293](#)
 - file_readwritable, [293](#)
 - file_temp_handle, [293](#)
 - file_world_accessible, [294](#)
 - folder_count, [294](#)
 - folder_exists, [294](#)
 - host_platform, [294](#)
 - host_version, [295](#)
 - ip_addr_eq, [295](#)
 - ip_addr_st, [295](#)
 - ip_copy, [296](#)

- ip_family, 296
- ip_localhost, 296
- ip_matches_subnet, 297
- ip_octet, 297
- ip_presentation, 297
- ip_private, 298
- ip_reversed, 298
- ip_segment, 298
- ip_standard, 299
- ip_subnet, 299
- ip_subnet_st, 299
- ip_type, 300
- ip_word, 300
- MAGMA_SPOOL_BASE, 287
- MAGMA_SPOOL_DATA, 287
- MAGMA_SPOOL_SCAN, 287
- MAGMA_COLOR_BLUE, 283
- MAGMA_COLOR_BLUE_BOLD, 283
- MAGMA_COLOR_BLUE_INTENSE, 283
- MAGMA_COLOR_BLUE_INTENSE_BOLD, 283
- MAGMA_COLOR_BLUE_UNDERLINE, 283
- MAGMA_COLOR_CYAN, 284
- MAGMA_COLOR_CYAN_BOLD, 284
- MAGMA_COLOR_CYAN_INTENSE, 284
- MAGMA_COLOR_CYAN_INTENSE_BOLD, 284
- MAGMA_COLOR_CYAN_UNDERLINE, 284
- MAGMA_COLOR_GREEN, 284
- MAGMA_COLOR_GREEN_BOLD, 284
- MAGMA_COLOR_GREEN_INTENSE, 284
- MAGMA_COLOR_GREEN_INTENSE_BOLD, 284
- MAGMA_COLOR_GREEN_UNDERLINE, 285
- MAGMA_COLOR_PURPLE, 285
- MAGMA_COLOR_PURPLE_BOLD, 285
- MAGMA_COLOR_PURPLE_INTENSE, 285
- MAGMA_COLOR_PURPLE_INTENSE_BOLD, 285
- MAGMA_COLOR_PURPLE_UNDERLINE, 285
- MAGMA_COLOR_RED, 285
- MAGMA_COLOR_RED_BOLD, 285
- MAGMA_COLOR_RED_INTENSE, 285
- MAGMA_COLOR_RED_INTENSE_BOLD, 286
- MAGMA_COLOR_RED_UNDERLINE, 286
- MAGMA_COLOR_RESET, 286
- MAGMA_COLOR_WHITE, 286
- MAGMA_COLOR_WHITE_BOLD, 286
- MAGMA_COLOR_WHITE_INTENSE, 286
- MAGMA_COLOR_WHITE_INTENSE_BOLD, 286
- MAGMA_COLOR_WHITE_UNDERLINE, 286
- MAGMA_COLOR_YELLOW, 286
- MAGMA_COLOR_YELLOW_BOLD, 287
- MAGMA_COLOR_YELLOW_INTENSE, 287
- MAGMA_COLOR_YELLOW_INTENSE_BOLD, 287
- MAGMA_COLOR_YELLOW_UNDERLINE, 287
- MAGMA_PROC_PATH, 287
- octet_t, 287
- process_find_pid, 300
- process_kill, 301
- process_my_pid, 301
- segment_t, 287
- signal_name, 301
- spool_check, 301
- spool_check_file, 302
- spool_cleanup, 302
- spool_error_stats, 302
- spool_mktemp, 303
- spool_path, 303
- spool_start, 303
- spool_stop, 304
- tcp_addr_ip, 304
- tcp_addr_st, 304
- tcp_continue, 304
- tcp_error, 304
- tcp_read, 305
- tcp_status, 305
- tcp_wait, 305
- tcp_write, 305
- host_platform
 - host.c, 278
 - host.h, 294
- host_version
 - host.c, 278
 - host.h, 295
- HTTP
 - engine/config/servers/servers.h, 753
- http
 - magma_t, 95
- http.c
 - http_body, 2110
 - http_close, 2110
 - http_init, 2111
 - http_process, 2111
 - http_requeue, 2111
- HTTP_CLOSE
 - servers/http/http.h, 855
- HTTP_COMPLETE
 - servers/http/http.h, 855
- HTTP_CONNECTION_CLOSE
 - servers/http/http.h, 855
- HTTP_CONNECTION_KEEPALIVE
 - servers/http/http.h, 855
- HTTP_CONNECTION_NEUTRAL
 - servers/http/http.h, 855
- HTTP_COOKIE_DELETE
 - servers/http/http.h, 855
- HTTP_COOKIE_NEUTRAL
 - servers/http/http.h, 855
- HTTP_COOKIE_SET
 - servers/http/http.h, 855
- HTTP_DATA_ANY
 - network/http.h, 850
- HTTP_DATA_GET
 - network/http.h, 850
- HTTP_DATA_HEADER
 - network/http.h, 850
- HTTP_DATA_POST

- network/http.h, 850
- HTTP_ERROR_400
 - servers/http/http.h, 855
- HTTP_ERROR_401
 - servers/http/http.h, 855
- HTTP_ERROR_403
 - servers/http/http.h, 855
- HTTP_ERROR_404
 - servers/http/http.h, 855
- HTTP_ERROR_405
 - servers/http/http.h, 855
- HTTP_ERROR_422
 - servers/http/http.h, 855
- HTTP_ERROR_500
 - servers/http/http.h, 855
- HTTP_ERROR_501
 - servers/http/http.h, 855
- HTTP_MERGED
 - network/http.h, 850
- HTTP_METHOD_CONNECT
 - network/http.h, 851
- HTTP_METHOD_DELETE
 - network/http.h, 850
- HTTP_METHOD_GET
 - network/http.h, 850
- HTTP_METHOD_HEAD
 - network/http.h, 850
- HTTP_METHOD_NONE
 - network/http.h, 850
- HTTP_METHOD_OPTIONS
 - network/http.h, 850
- HTTP_METHOD_POST
 - network/http.h, 850
- HTTP_METHOD_PUT
 - network/http.h, 850
- HTTP_METHOD_TRACE
 - network/http.h, 850
- HTTP_METHOD_UNSUPPORTED
 - network/http.h, 851
- HTTP_OK
 - servers/http/http.h, 855
- HTTP_PARSE_COOKIE
 - servers/http/http.h, 855
- HTTP_PARSE_HEADER
 - servers/http/http.h, 855
- HTTP_PARSE_PAIRS
 - servers/http/http.h, 855
- HTTP_PORTAL
 - network/http.h, 850
- HTTP_READ_BODY
 - servers/http/http.h, 855
- HTTP_READY
 - servers/http/http.h, 855
- HTTP_RESPOND
 - servers/http/http.h, 855
- http_body
 - http.c, 2110
- servers/http/http.h, 856
- http_close
 - http.c, 2110
 - servers/http/http.h, 857
- http_content_load_directory
 - content.c, 2106
 - servers/http/http.h, 857
- http_content_load_fonts
 - content.c, 2106
 - servers/http/http.h, 857
- http_content_refresh
 - content.c, 2106
 - servers/http/http.h, 857
- http_content_start
 - content.c, 2106
 - servers/http/http.h, 858
- http_content_stop
 - content.c, 2107
 - servers/http/http.h, 858
- http_content_t, 75
 - location, 75
 - next, 75
 - resource, 75
 - type, 75
- HTTP_DATA
 - network/http.h, 850
- http_data_free
 - servers/http/data.c, 500
 - servers/http/http.h, 858
- http_data_get
 - servers/http/data.c, 500
 - servers/http/http.h, 859
- http_data_header_parse
 - servers/http/data.c, 501
 - servers/http/http.h, 859
- http_data_header_parse_line
 - servers/http/data.c, 501
 - servers/http/http.h, 859
- http_data_t, 76
 - name, 76
 - source, 76
 - value, 76
- http_data_value_decode
 - servers/http/data.c, 501
 - servers/http/http.h, 860
- http_data_value_parse
 - servers/http/data.c, 502
 - servers/http/http.h, 860
- http_free_content
 - content.c, 2107
 - servers/http/http.h, 860
- http_get_static
 - content.c, 2107
 - servers/http/http.h, 861
- http_get_template
 - content.c, 2108
 - servers/http/http.h, 861

- http_init
 - http.c, 2111
 - servers/http/http.h, 861
- http_load_file
 - content.c, 2108
 - servers/http/http.h, 862
- http_method_t
 - network/http.h, 850
- http_page_free
 - content.c, 2108
 - servers/http/http.h, 862
- http_page_get
 - content.c, 2109
 - servers/http/http.h, 862
- http_page_t, 77
 - content, 77
 - doc_ctx, 77
 - doc_obj, 77
 - xpath_ctx, 77
- http_parse_context
 - servers/http/http.h, 863
 - servers/http/parse.c, 2085
- http_parse_header
 - servers/http/http.h, 863
 - servers/http/parse.c, 2085
- http_parse_method
 - servers/http/http.h, 864
 - servers/http/parse.c, 2086
- http_parse_origin
 - servers/http/http.h, 864
 - servers/http/parse.c, 2086
- http_parse_pairs
 - servers/http/http.h, 865
 - servers/http/parse.c, 2086
- http_print_301
 - servers/http/errors.c, 270
 - servers/http/http.h, 865
- http_print_400
 - servers/http/errors.c, 271
 - servers/http/http.h, 865
- http_print_403
 - servers/http/errors.c, 271
 - servers/http/http.h, 866
- http_print_404
 - servers/http/errors.c, 271
 - servers/http/http.h, 866
- http_print_405
 - servers/http/errors.c, 271
 - servers/http/http.h, 866
- http_print_500
 - servers/http/errors.c, 272
 - servers/http/http.h, 867
- http_print_500_log
 - servers/http/errors.c, 272
 - servers/http/http.h, 867
- http_print_501
 - servers/http/errors.c, 272
 - servers/http/http.h, 867
- http_process
 - http.c, 2111
 - servers/http/http.h, 867
- http_requeue
 - http.c, 2111
 - servers/http/http.h, 868
- http_response
 - response.c, 2112
 - servers/http/http.h, 868
- http_response_allow_cross
 - response.c, 2112
 - servers/http/http.h, 869
- http_response_connection
 - response.c, 2113
 - servers/http/http.h, 869
- http_response_cookie
 - response.c, 2113
 - servers/http/http.h, 869
- http_response_header
 - response.c, 2113
 - servers/http/http.h, 869
- http_response_options
 - response.c, 2114
 - servers/http/http.h, 870
- http_response_status
 - response.c, 2114
 - servers/http/http.h, 870
- http_session_destroy
 - servers/http/http.h, 870
 - servers/http/sessions.c, 1238
- http_session_reset
 - servers/http/http.h, 871
 - servers/http/sessions.c, 1238
- hvf_input
 - register_session_t, 123
- hvf_value
 - register_session_t, 123
- i16
 - multi_t, 112
- i2d_ECDSA_SIG_d
 - symbols.h, 2040
- i2d_ECPrivateKey_d
 - symbols.c, 1996
 - symbols.h, 2040
- i2d_OCSP_CERTID_d
 - symbols.c, 1997
 - symbols.h, 2040
- i2d_OCSP_RESPONSE_d
 - symbols.h, 2040
- i2d_X509_d
 - symbols.c, 1997
 - symbols.h, 2040
- i2o_ECPrivateKey_d
 - symbols.c, 1997
 - symbols.h, 2040

- i32
 - multi_t, 112
- i64
 - multi_t, 112
- i8
 - multi_t, 112
- id
 - __attribute__, 18
 - cached_object, 32
 - smtp_message_t, 147
- id_author
 - crypto.h, 1582
- id_destination
 - crypto.h, 1582
- id_origin
 - crypto.h, 1582
- id_recipient
 - crypto.h, 1582
- id_offset
 - signet_field_t, 136
- ident_mt_mt
 - multi.c, 511
 - strings.h, 544
- iface
 - magma_t, 95
- images.h
 - lib_load_freetype, 1834
 - lib_load_gd, 1834
 - lib_load_jpeg, 1834
 - lib_load_png, 1835
 - lib_version_freetype, 1835
 - lib_version_gd, 1835
 - lib_version_jpeg, 1835
 - lib_version_png, 1836
- IMAP
 - engine/config/servers/servers.h, 753
- imap
 - magma_t, 95
- imap.c
 - imap_append, 2124
 - imap_capability, 2124
 - imap_check, 2124
 - imap_close, 2124
 - imap_copy, 2124
 - imap_create, 2124
 - imap_delete, 2125
 - imap_examine, 2125
 - imap_expunge, 2125
 - imap_fetch, 2125
 - imap_id, 2126
 - imap_idle, 2126
 - imap_init, 2126
 - imap_invalid, 2126
 - imap_list, 2127
 - imap_login, 2127
 - imap_logout, 2127
 - imap_lsub, 2127
 - imap_noop, 2128
 - imap_rename, 2128
 - imap_search, 2128
 - imap_select, 2128
 - imap_starttls, 2128
 - imap_status, 2128
 - imap_store, 2129
 - imap_subscribe, 2129
 - imap_unsubscribe, 2129
- imap_append
 - imap.c, 2124
 - servers/imap/imap.h, 879
- imap_append_message
 - servers/imap/imap.h, 879
 - servers/imap/messages.c, 1090
- IMAP_ARGUMENT_TYPE_ARRAY
 - servers/imap/imap.h, 877
- IMAP_ARGUMENT_TYPE_ASTRING
 - servers/imap/imap.h, 877
- IMAP_ARGUMENT_TYPE_EMPTY
 - servers/imap/imap.h, 877
- IMAP_ARGUMENT_TYPE_LITERAL
 - servers/imap/imap.h, 877
- IMAP_ARGUMENT_TYPE_NSTRING
 - servers/imap/imap.h, 877
- IMAP_ARGUMENT_TYPE_QSTRING
 - servers/imap/imap.h, 877
- imap_arguments_t
 - network/imap.h, 872
- IMAP_ARRAY_RECURSION_LIMIT
 - servers/imap/imap.h, 877
- imap_build_array
 - output.c, 2130
 - servers/imap/imap.h, 879
- imap_build_array_isliteral
 - output.c, 2130
 - servers/imap/imap.h, 879
- imap_capability
 - imap.c, 2124
 - servers/imap/imap.h, 880
- imap_check
 - imap.c, 2124
 - servers/imap/imap.h, 880
- imap_close
 - imap.c, 2124
 - servers/imap/imap.h, 880
- imap_command_log_safe
 - servers/imap/parse.c, 2088
- imap_command_parser
 - servers/imap/imap.h, 880
 - servers/imap/parse.c, 2088
- imap_commands
 - servers/imap/commands.h, 1633
- imap_compare
 - servers/imap/commands.c, 1616
 - servers/imap/imap.h, 880
- imap_copy

- imap.c, 2124
- servers/imap/imap.h, 881
- imap_count_folder_levels
 - servers/imap/folders.c, 1065
 - servers/imap/imap.h, 881
- imap_create
 - imap.c, 2124
 - servers/imap/imap.h, 881
- imap_delete
 - imap.c, 2125
 - servers/imap/imap.h, 882
- imap_duplicate_messages
 - fetch.c, 2115
 - servers/imap/imap.h, 882
- imap_examine
 - imap.c, 2125
 - servers/imap/imap.h, 882
- imap_expunge
 - imap.c, 2125
 - servers/imap/imap.h, 883
- imap_fetch
 - imap.c, 2125
 - servers/imap/imap.h, 883
- imap_fetch_body
 - fetch.c, 2115
 - servers/imap/imap.h, 883
- IMAP_FETCH_BODY_HEADER
 - servers/imap/imap.h, 877
- imap_fetch_body_header
 - fetch.c, 2116
 - servers/imap/imap.h, 883
- IMAP_FETCH_BODY_HEADER_FIELDS
 - servers/imap/imap.h, 878
- IMAP_FETCH_BODY_HEADER_FIELDS_NOT
 - servers/imap/imap.h, 878
- IMAP_FETCH_BODY_MIME
 - servers/imap/imap.h, 878
- imap_fetch_body_mime
 - fetch.c, 2116
 - servers/imap/imap.h, 883
- IMAP_FETCH_BODY_PART
 - servers/imap/imap.h, 878
- imap_fetch_body_part
 - fetch.c, 2116
 - servers/imap/imap.h, 884
- imap_fetch_body_portion
 - fetch.c, 2116
 - servers/imap/imap.h, 884
- imap_fetch_body_tag
 - fetch.c, 2116
 - servers/imap/imap.h, 884
- IMAP_FETCH_BODY_TEXT
 - servers/imap/imap.h, 878
- imap_fetch_bodystucture
 - fetch.c, 2116
 - servers/imap/imap.h, 884
- imap_fetch_dataitems_t, 78
- body, 78
- bodystucture, 78
- envelope, 78
- flags, 78
- internaldate, 78
- normal, 79
- normal_partial, 79
- peek, 79
- peek_partial, 79
- rfc822, 79
- rfc822_header, 79
- rfc822_size, 79
- rfc822_text, 79
- uid, 79
- imap_fetch_envelope
 - fetch.c, 2117
 - servers/imap/imap.h, 884
- imap_fetch_free_items
 - fetch.c, 2117
 - servers/imap/imap.h, 884
- imap_fetch_message
 - fetch.c, 2117
 - servers/imap/imap.h, 885
- imap_fetch_parse_partial
 - fetch.c, 2117
 - servers/imap/imap.h, 885
- imap_fetch_response_add
 - fetch_response.c, 2120
 - servers/imap/imap.h, 885
- imap_fetch_response_free
 - fetch_response.c, 2120
 - servers/imap/imap.h, 885
- imap_fetch_response_t, 81
 - key, 81
 - next, 81
 - value, 81
- imap_fetch_return_header
 - fetch.c, 2117
 - servers/imap/imap.h, 885
- imap_fetch_return_message
 - fetch.c, 2118
 - servers/imap/imap.h, 885
- imap_fetch_return_mime
 - fetch.c, 2118
 - servers/imap/imap.h, 886
- imap_fetch_return_text
 - fetch.c, 2118
 - servers/imap/imap.h, 886
- imap_flag_action
 - servers/imap/flags.c, 2121
 - servers/imap/imap.h, 886
- IMAP_FLAG_ADD
 - servers/imap/imap.h, 878
- imap_flag_parse
 - servers/imap/flags.c, 2121
 - servers/imap/imap.h, 886
- IMAP_FLAG_REMOVE

- servers/imap/imap.h, 878
- IMAP_FLAG_REPLACE
 - servers/imap/imap.h, 878
- IMAP_FLAG_SILENT
 - servers/imap/imap.h, 878
- imap_folder_compare
 - servers/imap/folders.c, 1065
 - servers/imap/imap.h, 886
- imap_folder_create
 - servers/imap/folders.c, 1066
 - servers/imap/imap.h, 886
- imap_folder_name_escaped
 - servers/imap/folders.c, 1066
 - servers/imap/imap.h, 887
- IMAP_FOLDER_RECURSION_LMIIT
 - servers/imap/imap.h, 878
- imap_folder_remove
 - servers/imap/folders.c, 1066
 - servers/imap/imap.h, 887
- imap_folder_rename
 - servers/imap/folders.c, 1067
 - servers/imap/imap.h, 888
- imap_folder_status
 - servers/imap/folders.c, 1067
 - servers/imap/imap.h, 888
- imap_folder_status_t, 82
 - first, 82
 - foldernum, 82
 - messages, 82
 - recent, 82
 - uidnext, 82
 - unseen, 82
- imap_get_ar_ar
 - servers/imap/imap.h, 889
 - servers/imap/parse.c, 2089
- imap_get_flag
 - servers/imap/flags.c, 2121
 - servers/imap/imap.h, 889
- imap_get_ptr
 - servers/imap/imap.h, 889
 - servers/imap/parse.c, 2089
- imap_get_st_ar
 - servers/imap/imap.h, 889
 - servers/imap/parse.c, 2089
- imap_get_type_ar
 - servers/imap/imap.h, 890
 - servers/imap/parse.c, 2090
- imap_id
 - imap.c, 2126
 - servers/imap/imap.h, 890
- imap_idle
 - imap.c, 2126
 - servers/imap/imap.h, 890
- imap_init
 - imap.c, 2126
 - servers/imap/imap.h, 890
- imap_invalid
 - imap.c, 2126
 - servers/imap/imap.h, 891
- imap_list
 - imap.c, 2127
 - servers/imap/imap.h, 891
- imap_login
 - imap.c, 2127
 - servers/imap/imap.h, 891
- imap_logout
 - imap.c, 2127
 - servers/imap/imap.h, 892
- imap_lsub
 - imap.c, 2127
 - servers/imap/imap.h, 892
- imap_message_copier
 - servers/imap/imap.h, 892
 - servers/imap/messages.c, 1090
- imap_message_expunge
 - servers/imap/imap.h, 892
 - servers/imap/messages.c, 1090
- imap_narrow_folders
 - servers/imap/folders.c, 1068
 - servers/imap/imap.h, 892
- imap_narrow_messages
 - fetch.c, 2118
 - servers/imap/imap.h, 893
- imap_next_folder_order
 - servers/imap/folders.c, 1068
 - servers/imap/imap.h, 893
- imap_noop
 - imap.c, 2128
 - servers/imap/imap.h, 893
- imap_parse_address
 - parse_address.c, 2131
 - servers/imap/imap.h, 893
- imap_parse_address_breaker
 - parse_address.c, 2131
 - servers/imap/imap.h, 893
- imap_parse_address_part
 - parse_address.c, 2131
 - servers/imap/imap.h, 894
- imap_parse_address_put
 - parse_address.c, 2131
 - servers/imap/imap.h, 894
- imap_parse_arguments
 - servers/imap/imap.h, 894
 - servers/imap/parse.c, 2090
- imap_parse_array
 - servers/imap/imap.h, 894
 - servers/imap/parse.c, 2091
- imap_parse_astring
 - servers/imap/imap.h, 895
 - servers/imap/parse.c, 2091
- imap_parse_dataitems
 - fetch.c, 2118
 - servers/imap/imap.h, 895
- imap_parse_literal

- servers/imap/imap.h, 896
- servers/imap/parse.c, 2092
- imap_parse_nstring
 - servers/imap/imap.h, 896
 - servers/imap/parse.c, 2092
- imap_parse_qstring
 - servers/imap/imap.h, 896
 - servers/imap/parse.c, 2092
- imap_process
 - servers/imap/commands.c, 1616
 - servers/imap/imap.h, 897
- imap_range_build
 - range.c, 2132
 - servers/imap/imap.h, 897
- imap_rename
 - imap.c, 2128
 - servers/imap/imap.h, 897
- imap_requeue
 - servers/imap/commands.c, 1617
 - servers/imap/imap.h, 897
- imap_search
 - imap.c, 2128
 - servers/imap/imap.h, 898
- imap_search_flag
 - servers/imap/imap.h, 898
 - servers/imap/search.c, 219
- imap_search_messages
 - servers/imap/imap.h, 898
 - servers/imap/search.c, 219
- imap_search_messages_body
 - servers/imap/imap.h, 898
 - servers/imap/search.c, 219
- imap_search_messages_date
 - servers/imap/imap.h, 898
 - servers/imap/search.c, 220
- imap_search_messages_date_compare
 - servers/imap/imap.h, 899
 - servers/imap/search.c, 220
- imap_search_messages_header
 - servers/imap/imap.h, 899
 - servers/imap/search.c, 220
- imap_search_messages_inner
 - servers/imap/imap.h, 899
 - servers/imap/search.c, 220
- imap_search_messages_range
 - servers/imap/imap.h, 899
 - servers/imap/search.c, 220
- imap_search_messages_size
 - servers/imap/imap.h, 899
 - servers/imap/search.c, 220
- imap_search_messages_text
 - servers/imap/imap.h, 899
 - servers/imap/search.c, 220
- IMAP_SEARCH_RECURSION_LIMIT
 - servers/imap/imap.h, 879
- imap_select
 - imap.c, 2128
- servers/imap/imap.h, 899
- imap_session_destroy
 - servers/imap/imap.h, 900
 - servers/imap/sessions.c, 1239
- imap_session_t
 - network/imap.h, 872
- imap_session_update
 - servers/imap/imap.h, 900
 - servers/imap/sessions.c, 1239
- imap_sort
 - servers/imap/commands.c, 1617
 - servers/imap/imap.h, 900
- imap_starttls
 - imap.c, 2128
 - servers/imap/imap.h, 900
- imap_status
 - imap.c, 2128
 - servers/imap/imap.h, 901
- imap_store
 - imap.c, 2129
 - servers/imap/imap.h, 901
- imap_subscribe
 - imap.c, 2129
 - servers/imap/imap.h, 901
- imap_unsubscribe
 - imap.c, 2129
 - servers/imap/imap.h, 901
- imap_update_flags
 - servers/imap/flags.c, 2121
 - servers/imap/imap.h, 901
- imap_valid_folder_name
 - servers/imap/folders.c, 1068
 - servers/imap/imap.h, 901
- imap_valid_sequence
 - fetch.c, 2118
 - servers/imap/imap.h, 902
- impersonate_user
 - magma_t, 95
- importance
 - smtp_outbound_prefs_t, 149
- in_prefs
 - smtp_session_t, 153
- inbox
 - smtp_inbound_prefs_t, 143
- increase_resource_limits
 - magma_t, 95
- indent
 - magma_t, 95
- indexes.c
 - meta_inx_find, 1209
 - meta_inx_remove, 1209
- indexes.h
 - __attribute__, 338
 - hashed_alloc, 338
 - inx_alloc, 338
 - inx_append, 339
 - inx_auto_read, 339

- inx_auto_unlock, 339
- inx_auto_write, 339
- inx_cleanup, 340
- inx_count, 340
- inx_cursor_alloc, 340
- inx_cursor_free, 341
- inx_cursor_key_active, 341
- inx_cursor_key_next, 342
- inx_cursor_reset, 342
- inx_cursor_t, 347
- inx_cursor_value_active, 342
- inx_cursor_value_next, 342
- inx_delete, 343
- inx_find, 343
- inx_free, 344
- inx_insert, 344
- inx_lock_read, 344
- inx_lock_write, 345
- inx_options, 345
- inx_replace, 345
- inx_serial, 346
- inx_t, 347
- inx_truncate, 346
- inx_unlock, 346
- linked_alloc, 346
- M_INX_HASHED, 338
- M_INX_LINKED, 338
- M_INX_LOCK_MANUAL, 338
- M_INX_TREE, 338
- MAGMA_INDEX, 338
- MAGMA_INDEX_OPTION, 337
- MAGMA_INDEX_TYPE, 337
- info.c
 - st_info_allocator, 503
 - st_info_layout, 503
 - st_info_opts, 504
 - st_info_type, 504
 - st_option_allocators, 504
 - st_option_flags, 505
 - st_option_layouts, 505
 - st_option_types, 505
- init
 - magnum_t, 95
- INITIALIZE_DNS
 - dns.c, 1732
- INSERT_CONTACT
 - queries.h, 2072
- INSERT_FOLDER
 - queries.h, 2072
- INSERT_MESSAGE
 - queries.h, 2072
- INSERT_MESSAGE_DUPLICATE
 - queries.h, 2072
- INSERT_MESSAGE_TAG
 - queries.h, 2073
- INSERT_OBJECT
 - queries.h, 2073
- INSERT_RECEIVING
 - queries.h, 2073
- INSERT_SIGNATURE
 - queries.h, 2073
- INSERT_TRANSMITTING
 - queries.h, 2073
- int16_clamp
 - clamp.c, 400
 - numbers.h, 425
- int16_conv_bl
 - numbers.c, 409
 - numbers.h, 425
- int16_conv_ns
 - numbers.c, 410
 - numbers.h, 426
- int16_conv_st
 - numbers.c, 410
 - numbers.h, 426
- int16_digits
 - digits.c, 404
 - numbers.h, 427
- INT24_MAX
 - core.h, 225
- INT24_MIN
 - core.h, 225
- int24_t
 - core.h, 229
- int32_clamp
 - clamp.c, 400
 - numbers.h, 427
- int32_conv_bl
 - numbers.c, 410
 - numbers.h, 427
- int32_conv_ns
 - numbers.c, 411
 - numbers.h, 427
- int32_conv_st
 - numbers.c, 411
 - numbers.h, 428
- int32_digits
 - digits.c, 404
 - numbers.h, 428
- int64_clamp
 - clamp.c, 401
 - numbers.h, 428
- int64_conv_bl
 - numbers.c, 411
 - numbers.h, 429
- int64_conv_ns
 - numbers.c, 412
 - numbers.h, 429
- int64_conv_st
 - numbers.c, 412
 - numbers.h, 429
- int64_digits
 - digits.c, 405
 - numbers.h, 430

- int8_clamp
 - clamp.c, [401](#)
 - numbers.h, [430](#)
- int8_conv_bl
 - numbers.c, [413](#)
 - numbers.h, [430](#)
- int8_conv_ns
 - numbers.c, [413](#)
 - numbers.h, [431](#)
- int8_conv_st
 - numbers.c, [413](#)
 - numbers.h, [431](#)
- int8_digits
 - digits.c, [405](#)
 - numbers.h, [432](#)
- int_no_get_2b
 - misc_pub.c, [1558](#)
- int_no_get_3b
 - misc_pub.c, [1558](#)
- int_no_get_4b
 - misc_pub.c, [1558](#)
- int_no_put_2b
 - misc_pub.c, [1558](#)
- int_no_put_3b
 - misc_pub.c, [1558](#)
- int_no_put_4b
 - misc_pub.c, [1559](#)
- int_t
 - core.h, [229](#)
- internal
 - cached_store_t, [35](#)
- internaldate
 - imap_fetch_dataitems_t, [78](#)
- inx.c
 - inx_alloc, [349](#)
 - inx_append, [349](#)
 - inx_auto_read, [349](#)
 - inx_auto_unlock, [349](#)
 - inx_auto_write, [350](#)
 - inx_cleanup, [350](#)
 - inx_count, [350](#)
 - inx_delete, [350](#)
 - inx_find, [351](#)
 - inx_free, [351](#)
 - inx_insert, [351](#)
 - inx_lock_read, [352](#)
 - inx_lock_write, [352](#)
 - inx_options, [352](#)
 - inx_replace, [353](#)
 - inx_serial, [353](#)
 - inx_truncate, [353](#)
 - inx_unlock, [353](#)
- inx_alloc
 - indexes.h, [338](#)
 - inx.c, [349](#)
- inx_append
 - indexes.h, [339](#)
- inx.c, [349](#)
- inx_auto_read
 - indexes.h, [339](#)
 - inx.c, [349](#)
- inx_auto_unlock
 - indexes.h, [339](#)
 - inx.c, [349](#)
- inx_auto_write
 - indexes.h, [339](#)
 - inx.c, [350](#)
- inx_cleanup
 - indexes.h, [340](#)
 - inx.c, [350](#)
- inx_count
 - indexes.h, [340](#)
 - inx.c, [350](#)
- inx_cursor_alloc
 - cursors.c, [326](#)
 - indexes.h, [340](#)
- inx_cursor_free
 - cursors.c, [327](#)
 - indexes.h, [341](#)
- inx_cursor_key_active
 - cursors.c, [327](#)
 - indexes.h, [341](#)
- inx_cursor_key_next
 - cursors.c, [327](#)
 - indexes.h, [342](#)
- inx_cursor_reset
 - cursors.c, [328](#)
 - indexes.h, [342](#)
- inx_cursor_t
 - indexes.h, [347](#)
- inx_cursor_value_active
 - cursors.c, [328](#)
 - indexes.h, [342](#)
- inx_cursor_value_next
 - cursors.c, [328](#)
 - indexes.h, [342](#)
- inx_delete
 - indexes.h, [343](#)
 - inx.c, [350](#)
- inx_find
 - indexes.h, [343](#)
 - inx.c, [351](#)
- inx_free
 - indexes.h, [344](#)
 - inx.c, [351](#)
- inx_insert
 - indexes.h, [344](#)
 - inx.c, [351](#)
- inx_lock_read
 - indexes.h, [344](#)
 - inx.c, [352](#)
- inx_lock_write
 - indexes.h, [345](#)
 - inx.c, [352](#)

- inx_options
 - indexes.h, 345
 - inx.c, 352
- inx_replace
 - indexes.h, 345
 - inx.c, 353
- inx_serial
 - indexes.h, 346
 - inx.c, 353
- inx_t
 - indexes.h, 347
- inx_truncate
 - indexes.h, 346
 - inx.c, 353
- inx_unlock
 - indexes.h, 346
 - inx.c, 353
- ip
 - ip_t, 83
- ip.c
 - ip_addr_eq, 308
 - ip_addr_st, 308
 - ip_copy, 308
 - ip_family, 308
 - ip_localhost, 309
 - ip_matches_subnet, 309
 - ip_octet, 309
 - ip_presentation, 310
 - ip_private, 310
 - ip_reversed, 310
 - ip_segment, 311
 - ip_standard, 311
 - ip_subnet, 311
 - ip_subnet_st, 312
 - ip_type, 312
 - ip_word, 312
- ip4
 - ip_t, 83
- ip6
 - ip_t, 83
- ip_addr_eq
 - host.h, 295
 - ip.c, 308
- ip_addr_st
 - host.h, 295
 - ip.c, 308
- ip_address_v4
 - smtp_session_t, 153
- ip_copy
 - host.h, 296
 - ip.c, 308
- ip_family
 - host.h, 296
 - ip.c, 308
- ip_localhost
 - host.h, 296
 - ip.c, 309
- ip_matches_subnet
 - host.h, 297
 - ip.c, 309
- ip_octet
 - host.h, 297
 - ip.c, 309
- ip_presentation
 - host.h, 297
 - ip.c, 310
- ip_private
 - host.h, 298
 - ip.c, 310
- IP_RANDOMIZER_POOL
 - checkers.h, 1261
- IP_RANDOMIZER_PUSH_MAX
 - checkers.h, 1261
- IP_RANDOMIZER_PUSH_MIN
 - checkers.h, 1261
- ip_reversed
 - host.h, 298
 - ip.c, 310
- ip_segment
 - host.h, 298
 - ip.c, 311
- ip_standard
 - host.h, 299
 - ip.c, 311
- ip_subnet
 - host.h, 299
 - ip.c, 311
- ip_subnet_st
 - host.h, 299
 - ip.c, 312
- ip_t, 83
 - family, 83
 - ip, 83
 - ip4, 83
 - ip6, 83
- ip_type
 - host.h, 300
 - ip.c, 312
- ip_v4_allocations_t
 - allocations.h, 1259
- ip_word
 - host.h, 300
 - ip.c, 312
- ipv6
 - server_t, 131
- is_buf_zeroed
 - misc_pub.c, 1559
- is_hex_digit
 - sds.c, 1694
- IS_ROOT_LABEL
 - dns.h, 1749
- is_sep
 - dnskey, 56
- is_zone

- dnskey, 57
- ISNULL
 - database.h, 1445
- items
 - queue.c, 785
 - secure.c, 391
- jansson_flags
 - helpers.c, 2174
- jansson_version_d
 - symbols.h, 2040
- JOINTED
 - strings.h, 543
- jpeg.c
 - lib_load_jpeg, 1837
 - lib_version_jpeg, 1837
- jpeg_version_d
 - symbols.c, 1997
 - symbols.h, 2040
- json.c
 - lib_load_jansson, 1839
 - lib_version_jansson, 1839
- JSON_RPC_2_ERROR_APPLICATION
 - portal.h, 2210
- JSON_RPC_2_ERROR_PARSE_CHAR
 - portal.h, 2210
- JSON_RPC_2_ERROR_PARSE_ENCODING
 - portal.h, 2210
- JSON_RPC_2_ERROR_PARSE_MALFORMED
 - portal.h, 2210
- JSON_RPC_2_ERROR_SERVER_INTERNAL
 - portal.h, 2210
- JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS
 - portal.h, 2210
- JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL
 - portal.h, 2210
- JSON_RPC_2_ERROR_SERVER_REQUEST
 - portal.h, 2210
- JSON_RPC_2_ERROR_SYSTEM
 - portal.h, 2210
- JSON_RPC_2_ERROR_TRANSPORT
 - portal.h, 2210
- json_api.c
 - json_api_dispatch, 2175
- json_api.h
 - api_endpoint_auth, 2176
 - api_endpoint_change_password, 2176
 - api_endpoint_delete_user, 2176
 - api_endpoint_register, 2176
 - api_error, 2176
 - api_response, 2177
 - json_api_dispatch, 2177
- json_api_dispatch
 - json_api.c, 2175
 - json_api.h, 2177
- json_array_append_d
 - symbols.h, 2040
- json_array_append_new_d
 - symbols.h, 2041
- json_array_clear_d
 - symbols.h, 2041
- json_array_d
 - symbols.h, 2041
- json_array_extend_d
 - symbols.h, 2041
- json_array_get_d
 - symbols.h, 2041
- json_array_insert_d
 - symbols.h, 2041
- json_array_insert_new_d
 - symbols.h, 2041
- json_array_remove_d
 - symbols.h, 2041
- json_array_set_d
 - symbols.h, 2041
- json_array_set_new_d
 - symbols.h, 2041
- json_array_size_d
 - symbols.h, 2041
- json_copy_d
 - symbols.h, 2041
- json_decref_d
 - symbols.h, 2041
- json_deep_copy_d
 - symbols.h, 2041
- json_delete_d
 - symbols.h, 2042
- json_dump_file_d
 - symbols.h, 2042
- json_dumpf_d
 - symbols.h, 2042
- json_dumps_d
 - symbols.h, 2042
- json_equal_d
 - symbols.h, 2042
- json_false_d
 - symbols.h, 2042
- json_incref_d
 - symbols.h, 2042
- json_integer_d
 - symbols.h, 2042
- json_integer_set_d
 - symbols.h, 2042
- json_integer_value_d
 - symbols.h, 2042
- json_load_file_d
 - symbols.h, 2042
- json_loadf_d
 - symbols.h, 2042
- json_loads_d
 - symbols.h, 2042
- json_null_d
 - symbols.h, 2042
- json_number_value_d

- symbols.h, 2042
- json_object_clear_d
 - symbols.h, 2042
- json_object_d
 - symbols.h, 2042
- json_object_del_d
 - symbols.h, 2042
- json_object_get_d
 - symbols.h, 2043
- json_object_iter_at_d
 - symbols.h, 2043
- json_object_iter_d
 - symbols.h, 2043
- json_object_iter_key_d
 - symbols.h, 2043
- json_object_iter_next_d
 - symbols.h, 2043
- json_object_iter_set_d
 - symbols.h, 2043
- json_object_iter_set_new_d
 - symbols.h, 2043
- json_object_iter_value_d
 - symbols.h, 2043
- json_object_set_d
 - symbols.h, 2043
- json_object_set_new_d
 - symbols.h, 2043
- json_object_set_new_nocheck_d
 - symbols.h, 2043
- json_object_set_nocheck_d
 - symbols.h, 2043
- json_object_size_d
 - symbols.h, 2043
- json_object_update_d
 - symbols.h, 2043
- json_pack_d
 - symbols.h, 2044
- json_pack_ex_d
 - symbols.h, 2044
- json_real_d
 - symbols.h, 2044
- json_real_set_d
 - symbols.h, 2044
- json_real_value_d
 - symbols.h, 2044
- json_set_alloc_funcs_d
 - symbols.h, 2044
- json_string_d
 - symbols.h, 2044
- json_string_nocheck_d
 - symbols.h, 2044
- json_string_set_d
 - symbols.h, 2044
- json_string_set_nocheck_d
 - symbols.h, 2044
- json_string_value_d
 - symbols.h, 2044
- json_true_d
 - symbols.h, 2044
- json_type_string_d
 - symbols.h, 2044
- json_unpack_d
 - symbols.h, 2044
- json_unpack_ex_d
 - symbols.h, 2044
- json_vpack_ex_d
 - symbols.h, 2045
- json_vunpack_ex_d
 - symbols.h, 2045
- keks.c
 - keks_alloc, 1886
 - keks_cleanup, 1886
 - keks_free, 1886
 - keks_get, 1886
 - keks_set, 1886
- keks_alloc
 - chunks.h, 1877
 - keks.c, 1886
- keks_cleanup
 - chunks.h, 1877
 - keks.c, 1886
- keks_free
 - chunks.h, 1877
 - keks.c, 1886
- keks_get
 - chunks.h, 1878
 - keks.c, 1886
- keks_set
 - chunks.h, 1878
 - keks.c, 1886
- key
 - auth_legacy_t, 22
 - auth_t, 25
 - imap_fetch_response_t, 81
 - magma_t, 96
 - signet_field_t, 136
- key_pair_t, 84
 - private, 84
 - public, 84
- keynum
 - teacher_data_t, 157
- keys
 - auth_stacie_t, 23
 - auth_t, 26
- keys/orgs.c
 - org_encrypted_key_get, 1863
 - org_encrypted_key_set, 1863
 - org_key_alloc, 1863
 - org_key_free, 1863
 - org_key_generate, 1863
 - org_key_get, 1864
 - org_key_length, 1864
 - org_key_set, 1864

- keys/users.c
 - user_encrypted_key_get, 1867
 - user_encrypted_key_set, 1867
 - user_key_alloc, 1867
 - user_key_free, 1867
 - user_key_generate, 1867
 - user_key_get, 1868
 - user_key_length, 1868
 - user_key_set, 1868
- KEYS_ORG_PRIVATE_ENC
 - signet/common.h, 1576
- KEYS_ORG_PRIVATE_POK
 - signet/common.h, 1576
- KEYS_ORG_PRIVATE_SOK
 - signet/common.h, 1576
- KEYS_TYPE_ERROR
 - providers/dime/signet/keys.h, 667
- KEYS_TYPE_ORG
 - providers/dime/signet/keys.h, 667
- KEYS_TYPE_USER
 - providers/dime/signet/keys.h, 667
- KEYS_USER_PRIVATE_ENC
 - signet/common.h, 1576
- KEYS_USER_PRIVATE_SIGN
 - signet/common.h, 1576
- KEYS_FID_MAX
 - signet/common.h, 1574
- KEYS_HEADER_SIZE
 - signet/common.h, 1574
- keys_org_t
 - signet/common.h, 1576
- keys_type_t
 - providers/dime/signet/keys.h, 667
- keys_user_t
 - signet/common.h, 1576
- keytag
 - dnskey, 57
 - ds, 59
- label
 - dmime_header_key_t, 45
 - dnskey, 57
 - ds, 60
 - smtp_inbound_filter_t, 139
- label_length
 - dmime_header_key_t, 45
- legacy
 - auth_t, 26
- legacy.c
 - auth_legacy, 1024
 - auth_legacy_alloc, 1024
 - auth_legacy_free, 1024
- len
 - sha_databuf_t, 133
- length
 - magma_t, 96
 - secure.c, 391
 - tok_state_t, 160
 - transaction.c, 1493
- length.c
 - st_avail_get, 506
 - st_avail_set, 506
 - st_length_get, 507
 - st_length_int, 508
 - st_length_set, 509
- length_true
 - secure.c, 391
- lib_load
 - providers.h, 1938
 - symbols.c, 1984
- lib_load_bzip
 - bzip.c, 1287
 - compress.h, 1299
- lib_load_cache
 - consumers.h, 1316
 - providers/consumers/cache.c, 623
- lib_load_clamav
 - checkers.h, 1265
 - clamav.c, 1271
- lib_load_dkim
 - checkers.h, 1265
 - dkim.c, 1279
- lib_load_dspam
 - checkers.h, 1266
 - dspam.c, 1281
- lib_load_freetype
 - freetype.c, 1832
 - images.h, 1834
- lib_load_gd
 - gd.c, 1833
 - images.h, 1834
- lib_load_jansson
 - json.c, 1839
 - providers/parsers/parsers.h, 458
- lib_load_jpeg
 - images.h, 1834
 - jpeg.c, 1837
- lib_load_lzo
 - compress.h, 1300
 - lzo.c, 1304
- lib_load_mysql
 - database.h, 1446
 - mysql.c, 1467
- lib_load_openssl
 - cryptography/cryptography.h, 1368
 - openssl.c, 1409
- lib_load_png
 - images.h, 1835
 - png.c, 1838
- lib_load_spf
 - checkers.h, 1266
 - spf.c, 1283
- lib_load_tokyo
 - storage.h, 1960

- tokyo.c, 1971
- lib_load_utf8proc
 - providers/parsers/parsers.h, 458
 - utf8.c, 1840
- lib_load_xml
 - providers/parsers/parsers.h, 458
 - xml.c, 1844
- lib_load_zlib
 - compress.h, 1300
 - zlib.c, 1306
- lib_magma
 - freetype.c, 1832
- lib_symbols
 - providers.h, 1939
- lib_unload
 - providers.h, 1939
 - symbols.c, 1984
- lib_version
 - mysql.c, 1470
- lib_version_bzip
 - bzip.c, 1287
 - compress.h, 1300
- lib_version_cache
 - consumers.h, 1316
 - providers/consumers/cache.c, 623
- lib_version_clamav
 - checkers.h, 1266
 - clamav.c, 1271
- lib_version_dkim
 - checkers.h, 1266
 - dkim.c, 1279
- lib_version_dspam
 - checkers.h, 1267
 - dspam.c, 1281
- lib_version_freetype
 - freetype.c, 1832
 - images.h, 1835
- lib_version_gd
 - gd.c, 1833
 - images.h, 1835
- lib_version_jansson
 - json.c, 1839
 - providers/parsers/parsers.h, 459
- lib_version_jpeg
 - images.h, 1835
 - jpeg.c, 1837
- lib_version_lzo
 - compress.h, 1300
 - lzo.c, 1304
- lib_version_mysql
 - database.h, 1446
 - mysql.c, 1467
- lib_version_openssl
 - cryptography/cryptography.h, 1368
 - openssl.c, 1409
- lib_version_png
 - images.h, 1836
 - png.c, 1838
- lib_version_spf
 - checkers.h, 1267
 - spf.c, 1283
- lib_version_tokyo
 - storage.h, 1960
 - tokyo.c, 1971
- lib_version_utf8proc
 - providers/parsers/parsers.h, 459
 - utf8.c, 1840
- lib_version_xml
 - providers/parsers/parsers.h, 459
 - xml.c, 1844
- lib_version_zlib
 - compress.h, 1301
 - zlib.c, 1306
- libdime_base64_decode
 - encoding.c, 1825
 - encoding.h, 1826
- libdime_base64_encode
 - encoding.c, 1825
 - encoding.h, 1826
- library
 - magma_t, 96
- line
 - magma_t, 96
- line.c
 - line_pl_bl, 398
 - line_pl_ns, 398
 - line_pl_pl, 399
 - line_pl_st, 399
- line_pl_bl
 - core/parsers/parsers.h, 444
 - line.c, 398
- line_pl_ns
 - core/parsers/parsers.h, 444
 - line.c, 398
- line_pl_pl
 - core/parsers/parsers.h, 444
 - line.c, 399
- line_pl_st
 - core/parsers/parsers.h, 445
 - line.c, 399
- linked.c
 - __attribute__, 356
 - linked_alloc, 356
 - linked_append, 356
 - linked_cursor_active, 357
 - linked_cursor_alloc, 357
 - linked_cursor_free, 358
 - linked_cursor_key_active, 358
 - linked_cursor_key_next, 358
 - linked_cursor_next, 358
 - linked_cursor_reset, 359
 - linked_cursor_t, 363
 - linked_cursor_value_active, 359
 - linked_cursor_value_next, 359

- linked_delete, 360
- linked_find, 360
- linked_free, 360
- linked_insert, 361
- linked_node_t, 363
- linked_record_alloc, 361
- linked_record_free, 361
- linked_record_get_data, 362
- linked_record_get_key, 362
- linked_record_t, 363
- linked_truncate, 362
- linked_alloc
 - indexes.h, 346
 - linked.c, 356
- linked_append
 - linked.c, 356
- linked_cursor_active
 - linked.c, 357
- linked_cursor_alloc
 - linked.c, 357
- linked_cursor_free
 - linked.c, 358
- linked_cursor_key_active
 - linked.c, 358
- linked_cursor_key_next
 - linked.c, 358
- linked_cursor_next
 - linked.c, 358
- linked_cursor_reset
 - linked.c, 359
- linked_cursor_t
 - linked.c, 363
- linked_cursor_value_active
 - linked.c, 359
- linked_cursor_value_next
 - linked.c, 359
- linked_delete
 - linked.c, 360
- linked_find
 - linked.c, 360
- linked_free
 - linked.c, 360
- linked_insert
 - linked.c, 361
- linked_node_t
 - linked.c, 363
- linked_record_alloc
 - linked.c, 361
- linked_record_free
 - linked.c, 361
- linked_record_get_data
 - linked.c, 362
- linked_record_get_key
 - linked.c, 362
- linked_record_t
 - linked.c, 363
- linked_truncate
 - linked.c, 362
- links
 - magma_t, 96
- lint
 - symbols.h, 2023
- listen_queue
 - server_config_t, 127
 - server_t, 131
- listeners.c
 - net_accept, 903
 - net_init, 903
 - net_listen, 903
 - net_shutdown, 903
 - net_trigger, 904
- load_cache_contents
 - cache_pub.c, 1709
- load_ec_privkey
 - crypto_pub.c, 1512
- load_ec_pubkey
 - crypto_pub.c, 1513
- load_ed25519_privkey
 - crypto_pub.c, 1513
- load_message.c
 - mail_load_header, 1109
 - mail_load_message, 1109
 - mail_load_message_top, 1110
- local_size
 - smtp_inbound_prefs_t, 143
- location
 - __attribute__, 18
 - http_content_t, 75
 - magma_t, 96
 - smtp_inbound_filter_t, 139
- lock
 - cached_store_t, 35
 - magma_t, 96
 - objects/sessions/sessions.c, 1236
 - queue.c, 785
 - secure.c, 391
- lock_get
 - locks.c, 1094
 - objects.h, 1225
- lock_release
 - locks.c, 1095
 - objects.h, 1225
- locked
 - auth_t, 26
- locking.c
 - meta_user_rlock, 1210
 - meta_user_unlock, 1210
 - meta_user_wlock, 1211
- locks
 - engine/status/statistics.c, 805
- locks.c
 - lock_get, 1094
 - lock_release, 1095
 - MAGMA_LOCK_EXPIRATION, 1094

- MAGMA_LOCK_TIMEOUT, 1094
- user_lock, 1095
- user_unlock, 1095
- log
 - magma_t, 96
- log.c
 - log_backtrace, 788
 - log_date, 790
 - log_descriptor, 790
 - log_disable, 788
 - log_enable, 788
 - log_enabled, 790
 - log_internal, 789
 - log_mutex, 790
 - log_rotate, 789
 - log_start, 789
- log.h
 - log_backtrace, 793
 - log_check, 791
 - log_critical, 791
 - log_disable, 793
 - log_enable, 793
 - log_error, 792
 - log_info, 792
 - log_internal, 793
 - log_options, 792
 - log_pedantic, 792
 - log_rotate, 794
 - log_start, 794
 - log_warn, 792
 - M_LOG_CONSOLE, 792
 - M_LOG_CRITICAL, 792
 - M_LOG_CRITICAL_DISABLE, 793
 - M_LOG_ERROR, 792
 - M_LOG_ERROR_DISABLE, 793
 - M_LOG_FILE, 792
 - M_LOG_FILE_DISABLE, 793
 - M_LOG_FUNCTION, 792
 - M_LOG_FUNCTION_DISABLE, 793
 - M_LOG_INFO, 792
 - M_LOG_INFO_DISABLE, 792
 - M_LOG_LINE, 792
 - M_LOG_LINE_DISABLE, 793
 - M_LOG_LINE_FEED_DISABLE, 793
 - M_LOG_PEDANTIC, 792
 - M_LOG_PEDANTIC_DISABLE, 792
 - M_LOG_STACK_TRACE, 792
 - M_LOG_STACK_TRACE_DISABLE, 793
 - M_LOG_TIME, 792
 - M_LOG_TIME_DISABLE, 793
 - M_LOG_WARN, 792
 - M_LOG_WARN_DISABLE, 793
 - M_LOG_OPTIONS, 792
 - MAGMA_LOG_LEVELS, 792
- LOG_CODE_DEBUG
 - dime_ctx.h, 1567
- LOG_CODE_ERROR
 - dime_ctx.h, 1567
- LOG_CODE_INFO
 - dime_ctx.h, 1567
- log_backtrace
 - log.c, 788
 - log.h, 793
- log_callback
 - dime_ctx, 37
- log_check
 - core.h, 225
 - log.h, 791
- log_code_t
 - dime_ctx.h, 1567
- log_critical
 - core.h, 225
 - log.h, 791
- log_date
 - log.c, 790
- log_descriptor
 - log.c, 790
 - magma.c, 821
- log_disable
 - log.c, 788
 - log.h, 793
- log_enable
 - log.c, 788
 - log.h, 793
- log_enabled
 - core.h, 231
 - log.c, 790
- log_error
 - core.h, 225
 - log.h, 792
- log_function_t
 - dime_ctx.h, 1567
- log_info
 - core.h, 226
 - log.h, 792
- log_internal
 - log.c, 789
 - log.h, 793
- LOG_LEVEL_DEBUG
 - dime_ctx.c, 1564
 - dime_ctx.h, 1568
- LOG_LEVEL_ERROR
 - dime_ctx.c, 1564
 - dime_ctx.h, 1568
- LOG_LEVEL_INFO
 - dime_ctx.c, 1564
 - dime_ctx.h, 1568
- log_level_t, 85
 - code, 85
 - name, 85
- log_mutex
 - core.h, 231
 - log.c, 790
 - xml.c, 1854

- LOG_OPENSSL_ERROR
 - encrypt.c, [1827](#)
- log_options
 - core.h, [226](#)
 - log.h, [792](#)
- log_pedantic
 - core.h, [226](#)
 - log.h, [792](#)
- log_rotate
 - log.c, [789](#)
 - log.h, [794](#)
- log_start
 - log.c, [789](#)
 - log.h, [794](#)
- log_warn
 - log.h, [792](#)
- LOGDIR
 - symbols.h, [2023](#)
- lower_chr
 - case.c, [392](#)
 - core/parsers/parsers.h, [445](#)
- lower_st
 - case.c, [392](#)
 - core/parsers/parsers.h, [445](#)
- lt_dlexit_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- lzo.c
 - compress_lzo, [1303](#)
 - decompress_block_lzo, [1303](#)
 - decompress_lzo, [1303](#)
 - lib_load_lzo, [1304](#)
 - lib_version_lzo, [1304](#)
- lzo1x_1_compress_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- lzo1x_decompress_safe_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- lzo_adler32_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- lzo_version_string_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- m
 - random_data_t, [122](#)
 - test_data_t, [159](#)
- M_FOLDER_CONTACTS
 - folders.h, [1075](#)
- M_FOLDER_MESSAGES
 - folders.h, [1075](#)
- M_INX_HASHED
 - indexes.h, [338](#)
- M_INX_LINKED
 - indexes.h, [338](#)
- M_INX_LOCK_MANUAL
 - indexes.h, [338](#)
- M_INX_TREE
 - indexes.h, [338](#)
- M_LOG_CONSOLE
 - log.h, [792](#)
- M_LOG_CRITICAL
 - log.h, [792](#)
- M_LOG_CRITICAL_DISABLE
 - log.h, [793](#)
- M_LOG_ERROR
 - log.h, [792](#)
- M_LOG_ERROR_DISABLE
 - log.h, [793](#)
- M_LOG_FILE
 - log.h, [792](#)
- M_LOG_FILE_DISABLE
 - log.h, [793](#)
- M_LOG_FUNCTION
 - log.h, [792](#)
- M_LOG_FUNCTION_DISABLE
 - log.h, [793](#)
- M_LOG_INFO
 - log.h, [792](#)
- M_LOG_INFO_DISABLE
 - log.h, [792](#)
- M_LOG_LINE
 - log.h, [792](#)
- M_LOG_LINE_DISABLE
 - log.h, [793](#)
- M_LOG_LINE_FEED_DISABLE
 - log.h, [793](#)
- M_LOG_PEDANTIC
 - log.h, [792](#)
- M_LOG_PEDANTIC_DISABLE
 - log.h, [792](#)
- M_LOG_STACK_TRACE
 - log.h, [792](#)
- M_LOG_STACK_TRACE_DISABLE
 - log.h, [793](#)
- M_LOG_TIME
 - log.h, [792](#)
- M_LOG_TIME_DISABLE
 - log.h, [793](#)
- M_LOG_WARN
 - log.h, [792](#)
- M_LOG_WARN_DISABLE
 - log.h, [793](#)
- M_TYPE_BLOCK
 - core.h, [230](#)
- M_TYPE_BOOLEAN
 - core.h, [230](#)
- M_TYPE_DOUBLE
 - core.h, [230](#)
- M_TYPE_EMPTY
 - core.h, [230](#)
- M_TYPE_ENUM
 - core.h, [230](#)

- core.h, [230](#)
- M_TYPE_FLOAT
 - core.h, [230](#)
- M_TYPE_INT16
 - core.h, [230](#)
- M_TYPE_INT32
 - core.h, [230](#)
- M_TYPE_INT64
 - core.h, [230](#)
- M_TYPE_INT8
 - core.h, [230](#)
- M_TYPE_MULTI
 - core.h, [230](#)
- M_TYPE_NULLER
 - core.h, [230](#)
- M_TYPE_PLACER
 - core.h, [230](#)
- M_TYPE_STRINGER
 - core.h, [230](#)
- M_TYPE_UINT16
 - core.h, [230](#)
- M_TYPE_UINT32
 - core.h, [230](#)
- M_TYPE_UINT64
 - core.h, [230](#)
- M_TYPE_UINT8
 - core.h, [230](#)
- M_BIND
 - symbols.h, [2023](#)
- M_LOG_OPTIONS
 - log.h, [792](#)
- M_POOL_STATUS
 - buckets.h, [169](#)
- M_PORT
 - engine/config/servers/servers.h, [753](#)
- M_PROTOCOL
 - engine/config/servers/servers.h, [753](#)
- M_SSL_BIO_NOCLOSE
 - servers/servers.h, [759](#)
- M_TYPE
 - core.h, [229](#)
- magma
 - global.c, [727](#)
 - magma.h, [823](#)
- magma.c
 - log_descriptor, [821](#)
 - main, [821](#)
- magma.h
 - bufLen, [823](#)
 - bufptr, [823](#)
 - magma, [823](#)
 - threadBuffer, [824](#)
- MAGMA_SPOOL_BASE
 - host.h, [287](#)
- MAGMA_SPOOL_DATA
 - host.h, [287](#)
- MAGMA_SPOOL_SCAN
 - host.h, [287](#)
- MAGMA_BLACKLIST_INSTANCES
 - engine/config/config.h, [673](#)
- MAGMA_CACHE_INSTANCES
 - engine/config/config.h, [673](#)
- MAGMA_CACHE_SERVER_RETRY
 - engine/config/config.h, [673](#)
- MAGMA_CACHE_SOCKET_TIMEOUT
 - engine/config/config.h, [674](#)
- MAGMA_CIPHERS_GENERIC
 - cryptography/cryptography.h, [1348](#)
- MAGMA_CIPHERS_HIGH
 - cryptography/cryptography.h, [1348](#)
- MAGMA_CIPHERS_LOW
 - cryptography/cryptography.h, [1348](#)
- MAGMA_CIPHERS_MEDIUM
 - cryptography/cryptography.h, [1349](#)
- MAGMA_COLOR_BLUE
 - host.h, [283](#)
- MAGMA_COLOR_BLUE_BOLD
 - host.h, [283](#)
- MAGMA_COLOR_BLUE_INTENSE
 - host.h, [283](#)
- MAGMA_COLOR_BLUE_INTENSE_BOLD
 - host.h, [283](#)
- MAGMA_COLOR_BLUE_UNDERLINE
 - host.h, [283](#)
- MAGMA_COLOR_CYAN
 - host.h, [284](#)
- MAGMA_COLOR_CYAN_BOLD
 - host.h, [284](#)
- MAGMA_COLOR_CYAN_INTENSE
 - host.h, [284](#)
- MAGMA_COLOR_CYAN_INTENSE_BOLD
 - host.h, [284](#)
- MAGMA_COLOR_CYAN_UNDERLINE
 - host.h, [284](#)
- MAGMA_COLOR_GREEN
 - host.h, [284](#)
- MAGMA_COLOR_GREEN_BOLD
 - host.h, [284](#)
- MAGMA_COLOR_GREEN_INTENSE
 - host.h, [284](#)
- MAGMA_COLOR_GREEN_INTENSE_BOLD
 - host.h, [284](#)
- MAGMA_COLOR_GREEN_UNDERLINE
 - host.h, [285](#)
- MAGMA_COLOR_PURPLE
 - host.h, [285](#)
- MAGMA_COLOR_PURPLE_BOLD
 - host.h, [285](#)
- MAGMA_COLOR_PURPLE_INTENSE
 - host.h, [285](#)
- MAGMA_COLOR_PURPLE_INTENSE_BOLD
 - host.h, [285](#)
- MAGMA_COLOR_PURPLE_UNDERLINE
 - host.h, [285](#)

MAGMA_COLOR_RED
host.h, [285](#)

MAGMA_COLOR_RED_BOLD
host.h, [285](#)

MAGMA_COLOR_RED_INTENSE
host.h, [285](#)

MAGMA_COLOR_RED_INTENSE_BOLD
host.h, [286](#)

MAGMA_COLOR_RED_UNDERLINE
host.h, [286](#)

MAGMA_COLOR_RESET
host.h, [286](#)

MAGMA_COLOR_WHITE
host.h, [286](#)

MAGMA_COLOR_WHITE_BOLD
host.h, [286](#)

MAGMA_COLOR_WHITE_INTENSE
host.h, [286](#)

MAGMA_COLOR_WHITE_INTENSE_BOLD
host.h, [286](#)

MAGMA_COLOR_WHITE_UNDERLINE
host.h, [286](#)

MAGMA_COLOR_YELLOW
host.h, [286](#)

MAGMA_COLOR_YELLOW_BOLD
host.h, [287](#)

MAGMA_COLOR_YELLOW_INTENSE
host.h, [287](#)

MAGMA_COLOR_YELLOW_INTENSE_BOLD
host.h, [287](#)

MAGMA_COLOR_YELLOW_UNDERLINE
host.h, [287](#)

MAGMA_CONNECTION_BUFFER_SIZE
engine/config/config.h, [674](#)

MAGMA_CORE_POOL_OBJECTS_LIMIT
buckets.h, [169](#)

MAGMA_CORE_POOL_TIMEOUT_LIMIT
buckets.h, [169](#)

MAGMA_CRYPTOGRAPHY_SEED_SIZE
engine/config/config.h, [674](#)

MAGMA_E2BIG
core/host/errors.c, [266](#)

MAGMA_EACCES
core/host/errors.c, [266](#)

MAGMA_EAGAIN
core/host/errors.c, [266](#)

MAGMA_EBADF
core/host/errors.c, [266](#)

MAGMA_EBUSY
core/host/errors.c, [266](#)

MAGMA_ECHILD
core/host/errors.c, [266](#)

MAGMA_EDOM
core/host/errors.c, [266](#)

MAGMA_EEXIST
core/host/errors.c, [266](#)

MAGMA_EFAULT
core/host/errors.c, [266](#)

MAGMA_EFBIG
core/host/errors.c, [266](#)

MAGMA_EINTR
core/host/errors.c, [267](#)

MAGMA_EINVAL
core/host/errors.c, [267](#)

MAGMA_EIO
core/host/errors.c, [267](#)

MAGMA_EISDIR
core/host/errors.c, [267](#)

MAGMA_EMFILE
core/host/errors.c, [267](#)

MAGMA_EMLINK
core/host/errors.c, [267](#)

MAGMA_ENFILE
core/host/errors.c, [267](#)

MAGMA_ENODEV
core/host/errors.c, [267](#)

MAGMA_ENOENT
core/host/errors.c, [267](#)

MAGMA_ENOEXEC
core/host/errors.c, [268](#)

MAGMA_ENOMEM
core/host/errors.c, [268](#)

MAGMA_ENOSPC
core/host/errors.c, [268](#)

MAGMA_ENOTBLK
core/host/errors.c, [268](#)

MAGMA_ENOTDIR
core/host/errors.c, [268](#)

MAGMA_ENOTTY
core/host/errors.c, [268](#)

MAGMA_ENXIO
core/host/errors.c, [268](#)

MAGMA_EPERM
core/host/errors.c, [268](#)

MAGMA_EPIPE
core/host/errors.c, [268](#)

MAGMA_ERANGE
core/host/errors.c, [269](#)

MAGMA_EROFS
core/host/errors.c, [269](#)

MAGMA_ESPIPE
core/host/errors.c, [269](#)

MAGMA_ESRCH
core/host/errors.c, [269](#)

MAGMA_ETXTBSY
core/host/errors.c, [269](#)

MAGMA_EXDEV
core/host/errors.c, [269](#)

MAGMA_FILENAME_MAX
engine/config/config.h, [674](#)

MAGMA_FILEPATH_MAX
engine/config/config.h, [674](#)

magma_folder_alloc
folders.h, [1077](#)

- objects/folders/folders.c, 1059
- magma_folder_children
 - folders.h, 1077
 - objects/folders/folders.c, 1059
- magma_folder_delete
 - folders.h, 1078
 - objects/folders/datatier.c, 691
- magma_folder_fetch
 - folders.h, 1078
 - objects/folders/datatier.c, 691
- magma_folder_find_full_name
 - folders.h, 1078
 - folders/find.c, 1057
- magma_folder_find_name
 - folders.h, 1079
 - folders/find.c, 1057
- magma_folder_find_number
 - folders.h, 1079
 - folders/find.c, 1058
- magma_folder_free
 - folders.h, 1079
 - objects/folders/folders.c, 1060
- magma_folder_funcs
 - folders.h, 1080
 - objects/folders/folders.c, 1060
- magma_folder_insert
 - folders.h, 1080
 - objects/folders/datatier.c, 692
- magma_folder_name
 - folders.h, 1080
 - objects/folders/folders.c, 1060
- magma_folder_rename
 - folders.h, 1081
 - objects/folders/datatier.c, 692
- magma_folder_t
 - folders.h, 1082
- MAGMA_HASHED_BUCKETS
 - hashed.c, 330
- MAGMA_HOSTNAME_MAX
 - engine/config/config.h, 674
- MAGMA_INDEX
 - indexes.h, 338
- MAGMA_INDEX_OPTION
 - indexes.h, 337
- MAGMA_INDEX_TYPE
 - indexes.h, 337
- magma_keys
 - engine/config/global/keys.h, 664
- magma_keys_t, 86
 - database, 86
 - description, 86
 - file, 86
 - name, 86
 - norm, 86
 - overwrite, 86
 - required, 87
 - set, 87

- store, 87
- MAGMA_LOCATION_CACHE
 - engine/config/config.h, 674
- MAGMA_LOCK_EXPIRATION
 - locks.c, 1094
- MAGMA_LOCK_TIMEOUT
 - locks.c, 1094
- MAGMA_LOG_LEVELS
 - log.h, 792
- MAGMA_LOGS
 - engine/config/config.h, 674
- MAGMA_PORTAL_VERSION
 - portal.h, 2209
- MAGMA_PROC_PATH
 - host.h, 287
- MAGMA_RELAY_INSTANCES
 - engine/config/config.h, 674
- MAGMA_RESOURCE_FONTS
 - engine/config/config.h, 674
- MAGMA_RESOURCE_LOCATION
 - engine/config/config.h, 674
- MAGMA_RESOURCE_PAGES
 - engine/config/config.h, 675
- MAGMA_RESOURCE_TEMPLATES
 - engine/config/config.h, 675
- MAGMA_RESOURCE_VIRUS
 - engine/config/config.h, 675
- MAGMA_SERVER_INSTANCES
 - engine/config/config.h, 675
- MAGMA_SMTP_LINE_WRAP_LENGTH
 - engine/config/config.h, 675
- MAGMA_SMTP_MAX_ADDRESS_SIZE
 - engine/config/config.h, 675
- MAGMA_SMTP_MAX_HELO_SIZE
 - engine/config/config.h, 675
- MAGMA_SMTP_MAX_MESSAGE_SIZE
 - engine/config/config.h, 675
- MAGMA_SMTP_RECIPIENT_LIMIT
 - engine/config/config.h, 675
- MAGMA_SMTP_RELAY_LIMIT
 - engine/config/config.h, 675
- magma_t, 88
 - abuse, 91
 - active, 91
 - address_length_limit, 91
 - admin, 91
 - allow_cross_domain, 91
 - available, 91
 - blacklists, 91
 - bypass_addr, 91
 - bypass_subnets, 92
 - cache, 92
 - close, 92
 - config, 92
 - connections, 92
 - contact, 92
 - content, 92

- core_dump_size_limit, 92
- count, 92, 93
- cryptography, 93
- daemonize, 93
- database, 93
- dhparams_large_keys, 93
- dhparams_rotate, 93
- dime, 93
- dkim, 93
- domain, 93
- enable, 93
- enable_core_dumps, 94
- enabled, 94
- file, 94
- fonts, 94
- function, 94
- helo_length_limit, 94
- host, 94, 95
- http, 95
- iface, 95
- imap, 95
- impersonate_user, 95
- increase_resource_limits, 95
- indent, 95
- init, 95
- key, 96
- length, 96
- library, 96
- line, 96
- links, 96
- location, 96
- lock, 96
- log, 96
- memory, 96
- message_length_limit, 96
- minimum_password_length, 96
- name, 97
- network_buffer, 97
- number, 97
- output, 97
- output_config, 97
- output_resource_limits, 97
- page_length, 97
- pages, 97
- password, 97
- path, 97
- pool, 98
- port, 98
- portal, 98
- premium, 98
- recipient_limit, 98
- registration, 98
- relay, 98
- relay_limit, 98
- retry, 98
- root, 98
- root_directory, 99
- safeguard, 99
- salt, 99
- schema, 99
- secure, 99
- seed_length, 99
- selector, 99
- sender, 99
- servers, 99
- session_timeout, 99
- sessions, 100
- signatures, 100
- signet, 100
- smtp, 100
- socket_path, 100
- spf, 100
- spool, 100
- stack, 100
- standard, 100
- statistics, 100
- storage, 101
- system, 101
- tank, 101
- templates, 101
- thread_stack_size, 101
- time, 101
- timeout, 101
- tls_redirect, 101
- unload, 101
- user, 102
- virus, 102
- web, 102
- worker_threads, 102
- wrap_line_length, 102
- MAGMA_THREAD_BUFFER_SIZE
 - engine/config/config.h, 675
- MAGMA_THREAD_STACK_SIZE
 - engine/config/config.h, 676
- MAGMA_WORKER_THREAD_LIMIT
 - engine/config/config.h, 676
- mail.c
 - portal_outbound_checks, 2199
 - portal_smtp_create_data, 2199
 - portal_smtp_merge_headers, 2200
 - portal_smtp_relay_message, 2200
- mail.h
 - mail_add_forward_headers, 1116
 - mail_add_inbound_headers, 1116
 - mail_add_outbound_headers, 1116
 - mail_add_required_headers, 1117
 - mail_build_signature, 1117
 - mail_cache_destroy, 1118
 - mail_cache_get, 1118
 - mail_cache_reset, 1118
 - mail_cache_set, 1119
 - mail_cache_start, 1119
 - mail_cache_stop, 1119
 - mail_cache_thread_stop, 1119

- mail_copy_message, 1120
- mail_count_received, 1120
- mail_create_directory, 1120
- mail_create_message, 1121
- mail_db_delete_message, 1121
- mail_db_hide_message, 1121
- mail_db_insert_duplicate_message, 1122
- mail_db_insert_message, 1122
- mail_db_update_message_folder, 1123
- mail_destroy, 1123
- mail_destroy_header, 1123
- mail_destroy_message, 1124
- mail_discover_encoding, 1124
- mail_discover_insertion_point, 1124
- mail_discover_type, 1125
- mail_domain_get, 1125
- mail_extract_address, 1126
- mail_extract_tag, 1126
- mail_get_boundary, 1126
- mail_get_chunk, 1127
- mail_header_end, 1127
- mail_header_fetch_all, 1127
- mail_header_fetch_cleaned, 1128
- mail_header_pop, 1128
- mail_headers, 1128
- mail_insert_chunk_base64, 1129
- mail_insert_chunk_text, 1129
- mail_load_header, 1130
- mail_load_message, 1130
- mail_load_message_top, 1131
- mail_message, 1131
- mail_message_cleanup, 1132
- mail_message_path, 1132
- mail_mime_boundary, 1133
- mail_mime_child, 1133
- mail_mime_content_encoding, 1133
- mail_mime_content_id, 1134
- mail_mime_count, 1134
- mail_mime_encode_part, 1134
- mail_mime_encoding, 1135
- mail_mime_free, 1135
- mail_mime_generate_boundary, 1135
- mail_mime_get_media_type, 1136
- mail_mime_get_smtp_envelope, 1136
- mail_mime_header, 1137
- mail_mime_part, 1137
- MAIL_MIME_RECURSION_LIMIT, 1115
- mail_mime_split, 1137
- mail_mime_type, 1138
- mail_mime_type_group, 1138
- mail_mime_type_parameters, 1138
- mail_mime_type_parameters_key, 1139
- mail_mime_type_parameters_value, 1139
- mail_mime_type_sub, 1139
- mail_mime_update, 1140
- mail_mod_subject, 1140
- mail_modify_part, 1141
- mail_move_message, 1141
- mail_path_finder, 1142
- mail_remove_message, 1142
- mail_signature_add, 1142
- MAIL_SIGNATURES_RECURSION_LIMIT, 1115
- mail_store_header, 1143
- mail_store_message, 1143
- mail_store_message_data, 1143
- MAIL_MARK_BLACKHOLED
 - objects/messages/messages.h, 1179
- MAIL_MARK_INFECTED
 - objects/messages/messages.h, 1179
- MAIL_MARK_JUNK
 - objects/messages/messages.h, 1179
- MAIL_MARK_PHISHING
 - objects/messages/messages.h, 1179
- MAIL_MARK_SPOOFED
 - objects/messages/messages.h, 1179
- MAIL_STATUS_ANSWERED
 - objects/messages/messages.h, 1179
- MAIL_STATUS_APPENDED
 - objects/messages/messages.h, 1179
- MAIL_STATUS_DELETED
 - objects/messages/messages.h, 1179
- MAIL_STATUS_DRAFT
 - objects/messages/messages.h, 1179
- MAIL_STATUS_EMPTY
 - objects/messages/messages.h, 1179
- MAIL_STATUS_ENCRYPTED
 - objects/messages/messages.h, 1179
- MAIL_STATUS_FLAGGED
 - objects/messages/messages.h, 1179
- MAIL_STATUS_HIDDEN
 - objects/messages/messages.h, 1179
- MAIL_STATUS_NONE
 - objects/messages/messages.h, 1179
- MAIL_STATUS_RECENT
 - objects/messages/messages.h, 1179
- MAIL_STATUS_SECURE
 - objects/messages/messages.h, 1179
- MAIL_STATUS_SEEN
 - objects/messages/messages.h, 1179
- MAIL_STATUS_TAGGED
 - objects/messages/messages.h, 1179
- mail_add_forward_headers
 - mail.h, 1116
 - objects/mail/headers.c, 1101
- mail_add_inbound_headers
 - mail.h, 1116
 - objects/mail/headers.c, 1102
- mail_add_outbound_headers
 - mail.h, 1116
 - objects/mail/headers.c, 1102
- mail_add_required_headers
 - mail.h, 1117
 - objects/mail/headers.c, 1103
- mail_build_signature

- mail.h, [1117](#)
- signatures.c, [1165](#)
- mail_cache_destroy
 - mail.h, [1118](#)
 - objects/mail/cache.c, [615](#)
- mail_cache_get
 - mail.h, [1118](#)
 - objects/mail/cache.c, [615](#)
- mail_cache_reset
 - mail.h, [1118](#)
 - objects/mail/cache.c, [616](#)
- mail_cache_set
 - mail.h, [1119](#)
 - objects/mail/cache.c, [616](#)
- mail_cache_start
 - mail.h, [1119](#)
 - objects/mail/cache.c, [616](#)
- mail_cache_stop
 - mail.h, [1119](#)
 - objects/mail/cache.c, [616](#)
- mail_cache_t, [103](#)
 - messagenum, [103](#)
 - text, [103](#)
- mail_cache_thread_stop
 - mail.h, [1119](#)
 - objects/mail/cache.c, [617](#)
- mail_copy_message
 - mail.h, [1120](#)
 - store_message.c, [1172](#)
- mail_count_received
 - mail.h, [1120](#)
 - objects/mail/counters.c, [1099](#)
- mail_create_directory
 - mail.h, [1120](#)
 - paths.c, [1162](#)
- mail_create_message
 - mail.h, [1121](#)
 - objects/mail/objects.c, [1155](#)
- mail_db_delete_message
 - mail.h, [1121](#)
 - objects/mail/datatier.c, [693](#)
- mail_db_hide_message
 - mail.h, [1121](#)
 - objects/mail/datatier.c, [693](#)
- mail_db_insert_duplicate_message
 - mail.h, [1122](#)
 - objects/mail/datatier.c, [694](#)
- mail_db_insert_message
 - mail.h, [1122](#)
 - objects/mail/datatier.c, [694](#)
- mail_db_update_message_folder
 - mail.h, [1123](#)
 - objects/mail/datatier.c, [695](#)
- mail_destroy
 - mail.h, [1123](#)
 - objects/mail/objects.c, [1155](#)
- mail_destroy_header
 - cleanup.c, [1097](#)
 - mail.h, [1123](#)
- mail_destroy_message
 - mail.h, [1124](#)
 - objects/mail/objects.c, [1156](#)
- mail_discover_encoding
 - mail.h, [1124](#)
 - signatures.c, [1166](#)
- mail_discover_insertion_point
 - mail.h, [1124](#)
 - signatures.c, [1166](#)
- mail_discover_type
 - mail.h, [1125](#)
 - signatures.c, [1167](#)
- mail_domain_get
 - mail.h, [1125](#)
 - parsing.c, [1161](#)
- mail_extract_address
 - mail.h, [1126](#)
 - parsing.c, [1161](#)
- mail_extract_tag
 - mail.h, [1126](#)
 - signatures.c, [1167](#)
- mail_get_boundary
 - mail.h, [1126](#)
 - signatures.c, [1167](#)
- mail_get_chunk
 - mail.h, [1127](#)
 - signatures.c, [1168](#)
- mail_header_end
 - mail.h, [1127](#)
 - objects/mail/counters.c, [1099](#)
- mail_header_fetch_all
 - mail.h, [1127](#)
 - objects/mail/headers.c, [1103](#)
- mail_header_fetch_cleaned
 - mail.h, [1128](#)
 - objects/mail/headers.c, [1103](#)
- mail_header_pop
 - mail.h, [1128](#)
 - objects/mail/headers.c, [1104](#)
- mail_headers
 - mail.h, [1128](#)
 - objects/mail/headers.c, [1104](#)
- mail_insert_chunk_base64
 - mail.h, [1129](#)
 - signatures.c, [1168](#)
- mail_insert_chunk_text
 - mail.h, [1129](#)
 - signatures.c, [1169](#)
- mail_load_header
 - load_message.c, [1109](#)
 - mail.h, [1130](#)
- mail_load_message
 - load_message.c, [1109](#)
 - mail.h, [1130](#)
- mail_load_message_top

- load_message.c, 1110
- mail.h, 1131
- mail_message
 - mail.h, 1131
 - objects/mail/objects.c, 1156
- mail_message_cleanup
 - cleanup.c, 1097
 - mail.h, 1132
- mail_message_path
 - mail.h, 1132
 - paths.c, 1162
- mail_message_t, 104
 - date, 104
 - from, 104
 - header_length, 104
 - mime, 104
 - subject, 104
 - text, 104
 - to, 105
- mail_mime_boundary
 - mail.h, 1133
 - mime.c, 1146
- mail_mime_child
 - mail.h, 1133
 - mime.c, 1146
- mail_mime_content_encoding
 - mail.h, 1133
 - mime.c, 1147
- mail_mime_content_id
 - mail.h, 1134
 - mime.c, 1147
- mail_mime_count
 - mail.h, 1134
 - mime.c, 1147
- mail_mime_encode_part
 - mail.h, 1134
 - mime.c, 1148
- mail_mime_encoding
 - mail.h, 1135
 - mime.c, 1148
- mail_mime_free
 - mail.h, 1135
 - mime.c, 1149
- mail_mime_generate_boundary
 - mail.h, 1135
 - mime.c, 1149
- mail_mime_get_media_type
 - mail.h, 1136
 - mime.c, 1149
- mail_mime_get_smtp_envelope
 - mail.h, 1136
 - mime.c, 1150
- mail_mime_header
 - mail.h, 1137
 - mime.c, 1150
- mail_mime_part
 - mail.h, 1137
- mime.c, 1150
- MAIL_MIME_RECURSION_LIMIT
 - mail.h, 1115
- mail_mime_split
 - mail.h, 1137
 - mime.c, 1151
- mail_mime_t, 106
 - body, 106
 - boundary, 106
 - children, 106
 - encoding, 106
 - entire, 106
 - header, 106
 - type, 107
- mail_mime_type
 - mail.h, 1138
 - mime.c, 1151
- mail_mime_type_group
 - mail.h, 1138
 - mime.c, 1152
- mail_mime_type_parameters
 - mail.h, 1138
 - mime.c, 1152
- mail_mime_type_parameters_key
 - mail.h, 1139
 - mime.c, 1152
- mail_mime_type_parameters_value
 - mail.h, 1139
 - mime.c, 1153
- mail_mime_type_sub
 - mail.h, 1139
 - mime.c, 1153
- mail_mime_update
 - mail.h, 1140
 - mime.c, 1153
- mail_mod_subject
 - mail.h, 1140
 - objects/mail/headers.c, 1104
- mail_modify_part
 - mail.h, 1141
 - signatures.c, 1169
- mail_move_message
 - mail.h, 1141
 - store_message.c, 1172
- mail_path_finder
 - mail.h, 1142
 - paths.c, 1162
- mail_remove_message
 - mail.h, 1142
 - remove_message.c, 1164
- mail_signature_add
 - mail.h, 1142
 - signatures.c, 1170
- MAIL_SIGNATURES_RECURSION_LIMIT
 - mail.h, 1115
- MAIL_STATUS_ALL_FLAGS
 - objects/messages/messages.h, 1177

- MAIL_STATUS_SYSTEM_FLAGS
 - objects/messages/messages.h, [1177](#)
- MAIL_STATUS_USER_FLAGS
 - objects/messages/messages.h, [1177](#)
- mail_store_header
 - mail.h, [1143](#)
 - objects/mail/headers.c, [1105](#)
- mail_store_message
 - mail.h, [1143](#)
 - store_message.c, [1173](#)
- mail_store_message_data
 - mail.h, [1143](#)
 - store_message.c, [1173](#)
- mailbox.c
 - pop_get_last, [2136](#)
 - pop_get_message, [2136](#)
 - pop_total_messages, [2137](#)
 - pop_total_size, [2137](#)
- mailfrom
 - smtp_session_t, [153](#)
- main
 - fuzz-curve25519.c, [1680](#)
 - fuzz-ed25519.c, [1682](#)
 - magma.c, [821](#)
 - test-internals.c, [1686](#)
 - test.c, [1689](#)
- maint
 - engine/context/process.c, [317](#)
- MANAGED
 - strings.h, [539](#)
- MANAGED_T
 - strings.h, [543](#)
- managed_t
 - strings.h, [578](#)
- MANAGEDBUF
 - strings.h, [539](#)
- MAPPED_T
 - strings.h, [543](#)
- mapped_t
 - strings.h, [578](#)
- mappings
 - encodings.h, [248](#)
 - mappings.c, [253](#)
- mappings.c
 - mappings, [253](#)
- mappings_t, [108](#)
 - base64, [108](#)
 - base64_mod, [108](#)
 - characters, [108](#)
 - values, [108](#)
 - zbase32, [108](#)
- mark
 - smtp_inbound_prefs_t, [143](#)
- mask
 - subnet_t, [156](#)
- master
 - auth_stacie_t, [23](#)
 - auth_t, [26](#)
- max_batch_size
 - ed25519-donna-batchverify.h, [1651](#)
- max_length
 - smtp_session_t, [153](#)
- maxticks
 - test-ticks.h, [1687](#)
- media_type_t, [109](#)
 - bin, [109](#)
 - extension, [109](#)
 - name, [109](#)
- media_types
 - mime.c, [1154](#)
- mem_append
 - misc_pub.c, [1559](#)
- memcached_add_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- memcached_append_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- memcached_behavior_set_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- memcached_cas_d
 - symbols.c, [1997](#)
 - symbols.h, [2045](#)
- memcached_create_d
 - symbols.c, [1998](#)
 - symbols.h, [2045](#)
- memcached_decrement_d
 - symbols.c, [1998](#)
 - symbols.h, [2045](#)
- memcached_decrement_with_initial_d
 - symbols.c, [1998](#)
 - symbols.h, [2046](#)
- memcached_delete_d
 - symbols.c, [1998](#)
 - symbols.h, [2046](#)
- memcached_flush_d
 - symbols.c, [1998](#)
 - symbols.h, [2046](#)
- memcached_free_d
 - symbols.c, [1998](#)
 - symbols.h, [2046](#)
- memcached_get_d
 - symbols.c, [1998](#)
 - symbols.h, [2046](#)
- memcached_increment_d
 - symbols.c, [1998](#)
 - symbols.h, [2046](#)
- memcached_increment_with_initial_d
 - symbols.c, [1998](#)
 - symbols.h, [2046](#)
- memcached_lib_version_d
 - symbols.c, [1998](#)
 - symbols.h, [2046](#)

- memcached_prepend_d
 - symbols.c, 1998
 - symbols.h, 2046
- memcached_replace_d
 - symbols.c, 1999
 - symbols.h, 2046
- memcached_server_add_with_weight_d
 - symbols.c, 1999
 - symbols.h, 2046
- memcached_set_d
 - symbols.c, 1999
 - symbols.h, 2046
- memcached_strerror_d
 - symbols.c, 1999
 - symbols.h, 2047
- memory
 - magma_t, 96
- memory.c
 - mm_alloc, 370
 - mm_cleanup_variadic, 371
 - mm_copy, 371
 - mm_dup, 372
 - mm_empty, 372
 - mm_free, 372
 - mm_move, 373
 - mm_set, 373
 - mm_wipe, 374
- memory.h
 - align, 378
 - bitwise_and, 378
 - bitwise_count, 378
 - bitwise_or, 378
 - bitwise_xor, 378
 - MEMORYBUF, 377
 - mm_alloc, 379
 - mm_cleanup, 377
 - mm_cleanup_variadic, 379
 - mm_copy, 380
 - mm_dup, 380
 - mm_empty, 380
 - mm_free, 381
 - mm_move, 381
 - mm_sec_alloc, 382
 - mm_sec_cleanup, 382
 - mm_sec_free, 382
 - MM_SEC_PAGE_ALIGNMENT_MIN, 377
 - MM_SEC_POOL_LENGTH_MIN, 378
 - mm_sec_realloc, 383
 - MM_SEC_REQUEST_ALIGNMENT, 378
 - mm_sec_secured, 383
 - mm_sec_start, 383
 - mm_sec_stats, 383
 - mm_sec_stop, 384
 - mm_set, 384
 - mm_wipe, 384
- memory/bitwise.c
 - bitwise_and, 365
 - bitwise_count, 365
 - bitwise_or, 365
 - bitwise_xor, 365
- MEMORYBUF
 - memory.h, 377
- merged
 - __attribute__, 18
- message
 - derror_t, 36
 - smtp_session_t, 153
- MESSAGE_CHUNK_STATE_CREATION
 - crypto.h, 1582
- MESSAGE_CHUNK_STATE_ENCODED
 - crypto.h, 1582
- MESSAGE_CHUNK_STATE_ENCRYPTED
 - crypto.h, 1582
- MESSAGE_CHUNK_STATE_NONE
 - crypto.h, 1582
- MESSAGE_CHUNK_STATE_SIGNED
 - crypto.h, 1582
- MESSAGE_CHUNK_STATE_UNKNOWN
 - crypto.h, 1582
- MESSAGE_STATE_AUTHOR_SIGNED
 - crypto.h, 1583
- MESSAGE_STATE_CHUNKS_SIGNED
 - crypto.h, 1582
- MESSAGE_STATE_COMPLETE
 - crypto.h, 1583
- MESSAGE_STATE_EMPTY
 - crypto.h, 1582
- MESSAGE_STATE_ENCODED
 - crypto.h, 1582
- MESSAGE_STATE_ENCRYPTED
 - crypto.h, 1582
- MESSAGE_STATE_INCOMPLETE
 - crypto.h, 1582
- MESSAGE_STATE_NONE
 - crypto.h, 1582
- message_alloc
 - objects/messages/messages.c, 1086
 - objects/messages/messages.h, 1179
- MESSAGE_ENCODING_7BIT
 - objects/messages/messages.h, 1177
- MESSAGE_ENCODING_8BIT
 - objects/messages/messages.h, 1177
- MESSAGE_ENCODING_BASE64
 - objects/messages/messages.h, 1177
- MESSAGE_ENCODING_QUOTED_PRINTABLE
 - objects/messages/messages.h, 1177
- MESSAGE_ENCODING_UNKNOWN
 - objects/messages/messages.h, 1178
- message_folder_alloc
 - folders.h, 1081
 - objects/folders/messages.c, 1084
- message_folder_create
 - folders.h, 1081
 - objects/folders/messages.c, 1084

- message_folder_free
 - folders.h, 1082
 - objects/folders/messages.c, 1084
- message_folder_remove
 - folders.h, 1082
 - objects/folders/messages.c, 1085
- message_folder_t
 - folders.h, 1074
- message_free
 - objects/messages/messages.c, 1086
 - objects/messages/messages.h, 1180
- MESSAGE_HEADER_SIZE
 - dmessage/common.h, 1570
- message_header_t
 - objects/messages/messages.h, 1185
- message_length_limit
 - magma_t, 96
- MESSAGE_LENGTH_SIZE
 - dmessage/common.h, 1570
- message_t
 - objects/messages/messages.h, 1185
- MESSAGE_TYPE_HTML
 - objects/messages/messages.h, 1178
- MESSAGE_TYPE_MULTI_ALTERNATIVE
 - objects/messages/messages.h, 1178
- MESSAGE_TYPE_MULTI_MIXED
 - objects/messages/messages.h, 1178
- MESSAGE_TYPE_MULTI_RELATED
 - objects/messages/messages.h, 1178
- MESSAGE_TYPE_MULTI_RFC822
 - objects/messages/messages.h, 1178
- MESSAGE_TYPE_MULTI_UNKOWN
 - objects/messages/messages.h, 1178
- MESSAGE_TYPE_PLAIN
 - objects/messages/messages.h, 1178
- MESSAGE_TYPE_UNKNOWN
 - objects/messages/messages.h, 1178
- messagenum
 - mail_cache_t, 103
 - smtp_inbound_prefs_t, 143
- messages
 - imap_folder_status_t, 82
- messages/meta.c
 - meta_message_by_number, 1188
 - meta_message_dupe, 1188
 - meta_message_free, 1189
 - meta_messages_copier, 1189
 - meta_messages_login_update, 1189
 - meta_messages_mover, 1190
 - meta_messages_update, 1190
 - meta_messages_update_sequences, 1191
- messages_update
 - objects/messages/messages.c, 1086
 - objects/messages/messages.h, 1180
- meta
 - object_cache_t, 116
- meta/meta.c
 - meta_alloc, 1192
 - meta_free, 1192
 - meta_get, 1192
- meta/serials.c
 - meta_user_serial_check, 1215
 - meta_user_serial_get, 1215
 - meta_user_serial_set, 1215
- META_BOUNCE
 - dmessage/common.h, 1571
- META_CHECKPOINT_FOLDERS
 - network/meta.h, 905
- META_CHECKPOINT_MESSAGES
 - network/meta.h, 905
- META_CHECKPOINT_USER
 - network/meta.h, 905
- META_GET_ALIASES
 - network/meta.h, 906
- META_GET_CONTACTS
 - network/meta.h, 906
- META_GET_FOLDERS
 - network/meta.h, 906
- META_GET_KEYS
 - network/meta.h, 906
- META_GET_MESSAGES
 - network/meta.h, 906
- META_GET_NONE
 - network/meta.h, 906
- META_LOCKED
 - network/meta.h, 906
- META_NEED_LOCK
 - network/meta.h, 906
- META_PROTOCOL_DMTP
 - network/meta.h, 906
- META_PROTOCOL_GENERIC
 - network/meta.h, 906
- META_PROTOCOL_IMAP
 - network/meta.h, 906
- META_PROTOCOL_JSON
 - network/meta.h, 906
- META_PROTOCOL_NONE
 - network/meta.h, 906
- META_PROTOCOL_POP
 - network/meta.h, 906
- META_PROTOCOL_SMTP
 - network/meta.h, 906
- META_PROTOCOL_WEB
 - network/meta.h, 906
- META_PROTOCOLCOL_DMAP
 - network/meta.h, 906
- META_USER_ADVERTISING
 - network/meta.h, 906
- META_USER_ENCRYPT_DATA
 - network/meta.h, 906
- META_USER_NONE
 - network/meta.h, 906
- META_USER_OVERQUOTA
 - network/meta.h, 906

META_USER_TLS
 network/meta.h, 906
 meta_alert_t
 network/meta.h, 907
 meta_alias_t
 network/meta.h, 907
 meta_alloc
 meta/meta.c, 1192
 objects/meta/meta.h, 911
 META_CHECKPOINT
 network/meta.h, 905
 meta_crypto_keys_create
 objects/meta/crypto.c, 1196
 objects/meta/meta.h, 912
 meta_data_acknowledge_alert
 objects/meta/datatier.c, 699
 objects/meta/meta.h, 912
 meta_data_delete_folder
 objects/meta/datatier.c, 699
 objects/meta/meta.h, 912
 meta_data_delete_tag
 objects/meta/datatier.c, 699
 objects/meta/meta.h, 913
 meta_data_fetch_alerts
 objects/meta/datatier.c, 700
 objects/meta/meta.h, 913
 meta_data_fetch_all_tags
 objects/meta/datatier.c, 700
 objects/meta/meta.h, 913
 meta_data_fetch_folder_messages
 objects/messages/datatier.c, 696
 objects/messages/messages.h, 1180
 meta_data_fetch_folders
 objects/meta/datatier.c, 700
 objects/meta/meta.h, 914
 meta_data_fetch_keys
 objects/meta/datatier.c, 701
 objects/meta/meta.h, 914
 meta_data_fetch_mailbox_aliases
 objects/meta/datatier.c, 701
 objects/meta/meta.h, 914
 meta_data_fetch_message_tags
 objects/messages/datatier.c, 696
 objects/messages/messages.h, 1181
 meta_data_fetch_messages
 objects/messages/datatier.c, 697
 objects/messages/messages.h, 1181
 meta_data_fetch_shard
 objects/meta/datatier.c, 701
 objects/meta/meta.h, 915
 meta_data_fetch_user
 objects/meta/datatier.c, 702
 objects/meta/meta.h, 915
 meta_data_flags_add
 objects/meta/datatier.c, 702
 objects/meta/meta.h, 916
 meta_data_flags_remove
 objects/meta/datatier.c, 703
 objects/meta/meta.h, 916
 meta_data_flags_replace
 objects/meta/datatier.c, 703
 objects/meta/meta.h, 916
 meta_data_insert_folder
 objects/meta/datatier.c, 703
 objects/meta/meta.h, 917
 meta_data_insert_keys
 objects/meta/datatier.c, 704
 objects/meta/meta.h, 917
 meta_data_insert_shard
 objects/meta/datatier.c, 704
 objects/meta/meta.h, 917
 meta_data_insert_tag
 objects/meta/datatier.c, 705
 objects/meta/meta.h, 918
 meta_data_truncate_tags
 objects/meta/datatier.c, 705
 objects/meta/meta.h, 918
 meta_data_update_folder_name
 objects/meta/datatier.c, 705
 objects/meta/meta.h, 918
 meta_data_update_lock
 objects/meta/meta.h, 919
 meta_data_update_log
 objects/meta/datatier.c, 706
 objects/meta/meta.h, 919
 META_FETCH_MAIL_KEYS
 queries.h, 2073
 META_FETCH_SHARD
 queries.h, 2073
 META_FETCH_USER
 queries.h, 2073
 meta_folder_stats_tag_alloc
 objects/meta/folders.c, 1062
 objects/meta/meta.h, 919
 meta_folder_t
 network/meta.h, 907
 meta_folders_by_name
 objects/meta/folders.c, 1062
 objects/meta/meta.h, 920
 meta_folders_by_number
 objects/meta/folders.c, 1063
 objects/meta/meta.h, 920
 meta_folders_children
 objects/meta/folders.c, 1063
 objects/meta/meta.h, 920
 meta_folders_name
 objects/meta/folders.c, 1063
 objects/meta/meta.h, 921
 meta_folders_stats_tags
 objects/meta/folders.c, 1064
 objects/meta/meta.h, 921
 meta_free
 meta/meta.c, 1192
 objects/meta/meta.h, 921

- META_GET
 - network/meta.h, 905
- meta_get
 - meta/meta.c, 1192
 - objects/meta/meta.h, 922
- META_INSERT_MAIL_KEYS
 - queries.h, 2073
- META_INSERT_SHARD
 - queries.h, 2073
- meta_inx_find
 - indexes.c, 1209
 - objects/meta/meta.h, 922
- meta_inx_remove
 - indexes.c, 1209
 - objects/meta/meta.h, 923
- META_LOCK_STATUS
 - network/meta.h, 906
- meta_message_by_number
 - messages/meta.c, 1188
 - objects/messages/messages.h, 1181
- meta_message_dupe
 - messages/meta.c, 1188
 - objects/messages/messages.h, 1182
- meta_message_free
 - messages/meta.c, 1189
 - objects/messages/messages.h, 1182
- meta_message_t
 - network/meta.h, 907
- meta_messages_copier
 - messages/meta.c, 1189
 - objects/messages/messages.h, 1182
- meta_messages_login_update
 - messages/meta.c, 1189
 - objects/messages/messages.h, 1183
- meta_messages_mover
 - messages/meta.c, 1190
 - objects/messages/messages.h, 1183
- meta_messages_update
 - messages/meta.c, 1190
 - objects/messages/messages.h, 1184
- meta_messages_update_sequences
 - messages/meta.c, 1191
 - objects/messages/messages.h, 1184
- META_PROTOCOL
 - network/meta.h, 906
- meta_stats_tag_t, 110
 - count, 110
 - tag, 110
- meta_update_aliases
 - objects/meta/meta.h, 923
 - updaters.c, 1220
- meta_update_contacts
 - objects/meta/meta.h, 923
 - updaters.c, 1220
- meta_update_folders
 - objects/meta/meta.h, 924
 - updaters.c, 1221
- meta_update_keys
 - objects/meta/meta.h, 924
 - updaters.c, 1221
- meta_update_message_folders
 - objects/meta/meta.h, 924
 - updaters.c, 1222
- meta_update_realms
 - objects/meta/meta.h, 925
 - updaters.c, 1222
- meta_update_user
 - objects/meta/meta.h, 925
 - updaters.c, 1222
- META_USER_FLAGS
 - network/meta.h, 906
- meta_user_ref_add
 - objects/meta/meta.h, 926
 - references.c, 1212
- meta_user_ref_dec
 - objects/meta/meta.h, 926
 - references.c, 1212
- meta_user_ref_protocol_total
 - objects/meta/meta.h, 927
 - references.c, 1213
- meta_user_ref_stamp
 - objects/meta/meta.h, 927
 - references.c, 1213
- meta_user_ref_total
 - objects/meta/meta.h, 927
 - references.c, 1213
- meta_user_rlock
 - locking.c, 1210
 - objects/meta/meta.h, 927
- meta_user_serial_check
 - meta/serials.c, 1215
 - objects/meta/meta.h, 928
- meta_user_serial_get
 - meta/serials.c, 1215
 - objects/meta/meta.h, 928
- meta_user_serial_set
 - meta/serials.c, 1215
 - objects/meta/meta.h, 928
- meta_user_t
 - network/meta.h, 907
- meta_user_unlock
 - locking.c, 1210
 - objects/meta/meta.h, 929
- meta_user_wlock
 - locking.c, 1211
 - objects/meta/meta.h, 929
- method
 - __attribute__, 19
- methods.h
 - portal_methods, 2202
- mime
 - mail_message_t, 104
- mime.c
 - mail_mime_boundary, 1146

- mail_mime_child, 1146
- mail_mime_content_encoding, 1147
- mail_mime_content_id, 1147
- mail_mime_count, 1147
- mail_mime_encode_part, 1148
- mail_mime_encoding, 1148
- mail_mime_free, 1149
- mail_mime_generate_boundary, 1149
- mail_mime_get_media_type, 1149
- mail_mime_get_smtp_envelope, 1150
- mail_mime_header, 1150
- mail_mime_part, 1150
- mail_mime_split, 1151
- mail_mime_type, 1151
- mail_mime_type_group, 1152
- mail_mime_type_parameters, 1152
- mail_mime_type_parameters_key, 1152
- mail_mime_type_parameters_value, 1153
- mail_mime_type_sub, 1153
- mail_mime_update, 1153
- media_types, 1154
- minimum_password_length
 - magma_t, 96
- MINIMUM_PAYLOAD_SIZE
 - dmessage/common.h, 1570
- misc.c
 - __dbgprint, 1538
 - __str_printf, 1538
 - _b64decode, 1538
 - _b64decode_nopad, 1539
 - _b64encode, 1539
 - _b64encode_nopad, 1539
 - _b64encode_w_lineseparators, 1540
 - _compute_crc24_checksum, 1540
 - _compute_sha_hash, 1540
 - _compute_sha_hash_multibuf, 1541
 - _count_ptr_chain, 1541
 - _dbgprint, 1541
 - _decode_rsa_pubkey, 1542
 - _dump_buf, 1542
 - _dump_buf_outer, 1542
 - _encode_rsa_pubkey, 1543
 - _get_chr_date, 1543
 - _get_dbg_level, 1543
 - _get_x509_cert_sha_hash, 1544
 - _hex_encode, 1544
 - _int_no_get_2b, 1544
 - _int_no_get_3b, 1545
 - _int_no_get_4b, 1545
 - _int_no_put_2b, 1545
 - _int_no_put_3b, 1545
 - _int_no_put_4b, 1545
 - _is_buf_zeroed, 1546
 - _mem_append, 1546
 - _ptr_chain_add, 1546
 - _ptr_chain_clone, 1547
 - _ptr_chain_free, 1547
 - _read_file_data, 1547
 - _read_pem_data, 1548
 - _secure_wipe, 1548
 - _set_dbg_level, 1548
 - _str_printf, 1548
 - _verbose, 1549
 - _write_pem_data, 1549
- misc.h
 - _verbose, 1554
 - ALERT_PRINT, 1551
 - ANSI_COLOR_RED, 1551
 - ANSI_COLOR_RESET, 1551
 - B64_DECODED_LEN, 1551
 - B64_ENCODED_LEN, 1551
 - chr_isprint, 1552
 - chr_isspace, 1552
 - PUBLIC_FUNC_DECL, 1554
 - PUBLIC_FUNC_DECL_VA, 1554
 - SHA_160_SIZE, 1552
 - SHA_256_SIZE, 1552
 - SHA_512_B64_SIZE, 1552
 - SHA_512_SIZE, 1552
- misc_pub.c
 - b64decode, 1556
 - b64decode_nopad, 1556
 - b64encode, 1556
 - b64encode_nopad, 1557
 - compute_sha_hash, 1557
 - compute_sha_hash_multibuf, 1557
 - count_ptr_chain, 1557
 - dbgprint, 1557
 - decode_rsa_pubkey, 1557
 - dump_buf, 1557
 - dump_buf_outer, 1557
 - encode_rsa_pubkey, 1557
 - get_chr_date, 1558
 - get_dbg_level, 1558
 - get_x509_cert_sha_hash, 1558
 - hex_encode, 1558
 - int_no_get_2b, 1558
 - int_no_get_3b, 1558
 - int_no_get_4b, 1558
 - int_no_put_2b, 1558
 - int_no_put_3b, 1558
 - int_no_put_4b, 1559
 - is_buf_zeroed, 1559
 - mem_append, 1559
 - ptr_chain_add, 1559
 - ptr_chain_clone, 1559
 - ptr_chain_free, 1559
 - read_file_data, 1559
 - read_pem_data, 1559
 - secure_wipe, 1559
 - set_dbg_level, 1560
 - str_printf, 1560
- MM_SEC_CHUNK_ALLOCATED
 - secure.c, 387

MM_SEC_CHUNK_AVAILABLE

secure.c, [387](#)

mm_alloc

memory.c, [370](#)

memory.h, [379](#)

mm_cleanup

memory.h, [377](#)

mm_cleanup_variadic

memory.c, [371](#)

memory.h, [379](#)

mm_cmp_ci_eq

compare.h, [209](#)

equal.c, [215](#)

mm_cmp_cs_eq

compare.h, [210](#)

equal.c, [215](#)

mm_copy

memory.c, [371](#)

memory.h, [380](#)

mm_dupe

memory.c, [372](#)

memory.h, [380](#)

mm_empty

memory.c, [372](#)

memory.h, [380](#)

mm_free

memory.c, [372](#)

memory.h, [381](#)

mm_move

memory.c, [373](#)

memory.h, [381](#)

mm_sec_alloc

memory.h, [382](#)

secure.c, [387](#)

mm_sec_chunk_merge

secure.c, [387](#)

mm_sec_chunk_new

secure.c, [387](#)

mm_sec_chunk_next

secure.c, [387](#)

mm_sec_chunk_prev

secure.c, [388](#)

mm_sec_cleanup

memory.h, [382](#)

secure.c, [388](#)

mm_sec_free

memory.h, [382](#)

secure.c, [388](#)

MM_SEC_PAGE_ALIGNMENT_MIN

memory.h, [377](#)

MM_SEC_POOL_LENGTH_MIN

memory.h, [378](#)

mm_sec_realloc

memory.h, [383](#)

secure.c, [389](#)

MM_SEC_REQUEST_ALIGNMENT

memory.h, [378](#)

mm_sec_secured

memory.h, [383](#)

secure.c, [389](#)

mm_sec_start

memory.h, [383](#)

secure.c, [389](#)

mm_sec_stats

memory.h, [383](#)

secure.c, [389](#)

mm_sec_stop

memory.h, [384](#)

secure.c, [390](#)

mm_set

memory.c, [373](#)

memory.h, [384](#)

mm_wipe

memory.c, [374](#)

memory.h, [384](#)

mode

__attribute__, [19](#)

dmtplib_session_t, [55](#)

modm-donna-32bit.h

bignum256modm, [1683](#)

bignum256modm_bits_per_limb, [1683](#)

bignum256modm_element_t, [1683](#)

bignum256modm_limb_size, [1683](#)

modm-donna-64bit.h

bignum256modm, [1684](#)

bignum256modm_bits_per_limb, [1684](#)

bignum256modm_element_t, [1684](#)

bignum256modm_limb_size, [1684](#)

MOLTEN

engine/config/servers/servers.h, [753](#)

molten.c

molten_init, [2133](#)

molten_invalid, [2133](#)

molten_quit, [2133](#)

molten_stats, [2133](#)

molten.h

molten_compare, [2134](#)

molten_init, [2134](#)

molten_invalid, [2134](#)

molten_parse, [2134](#)

molten_quit, [2134](#)

molten_session_destroy, [2135](#)

molten_sort, [2135](#)

molten_stats, [2135](#)

molten_commands

servers/molten/commands.h, [1634](#)

molten_compare

molten.h, [2134](#)

servers/molten/commands.c, [1618](#)

molten_init

molten.c, [2133](#)

molten.h, [2134](#)

molten_invalid

molten.c, [2133](#)

- molten.h, 2134
- molten_parse
 - molten.h, 2134
 - servers/molten/commands.c, 1618
- molten_quit
 - molten.c, 2133
 - molten.h, 2134
- molten_session_destroy
 - molten.h, 2135
 - servers/molten/sessions.c, 1240
- molten_sort
 - molten.h, 2135
 - servers/molten/commands.c, 1618
- molten_stats
 - molten.c, 2133
 - molten.h, 2135
- MONTH_LOOKUP
 - servers/imap/search.c, 221
- mrec.c
 - _deserialize_dime_record_cb, 1760
 - _destroy_dime_record, 1761
 - _destroy_dime_record_cb, 1761
 - _dump_dime_record_cb, 1761
 - _get_dime_record, 1761
 - _get_dime_record_from_file, 1762
 - _parse_dime_record, 1762
 - _serialize_dime_record_cb, 1763
 - _validate_dime_record, 1763
- mrec.h
 - _deserialize_dime_record_cb, 1765
 - _destroy_dime_record_cb, 1766
 - _dump_dime_record_cb, 1766
 - _serialize_dime_record_cb, 1766
 - dime_msg_policy, 1765
 - DIME_RECORD_DNS_PREFIX, 1765
 - dime_sub_policy, 1765
 - DIME_VERSION_NO, 1765
 - msg_experimental, 1765
 - msg_mixed, 1765
 - msg_strict, 1765
 - PUBLIC_FUNC_DECL, 1767
 - sub_explicit, 1765
 - sub_relaxed, 1765
 - sub_strict, 1765
- mrec_pub.c
 - destroy_dime_record, 1768
 - get_dime_record, 1768
 - get_dime_record_from_file, 1768
 - parse_dime_record, 1768
 - validate_dime_record, 1768
- msg_experimental
 - mrec.h, 1765
- msg_mixed
 - mrec.h, 1765
- msg_strict
 - mrec.h, 1765
- mt_dupe
- multi.c, 511
 - strings.h, 544
- mt_free
 - multi.c, 512
 - strings.h, 545
- mt_get_char
 - multi.c, 512
 - strings.h, 545
- mt_get_length
 - multi.c, 512
 - strings.h, 545
- mt_get_null
 - multi.c, 513
 - strings.h, 546
- mt_get_number
 - multi.c, 513
 - strings.h, 546
- mt_get_type
 - multi.c, 513
 - strings.h, 546
- mt_is_empty
 - multi.c, 514
 - strings.h, 547
- mt_is_number
 - multi.c, 514
 - strings.h, 547
- mt_set_type
 - multi.c, 514
 - strings.h, 547
- mul32x32_64
 - ed25519-donna-portable.h, 1656
- multi.c
 - cmp_mt_mt, 510
 - ident_mt_mt, 511
 - mt_dupe, 511
 - mt_free, 512
 - mt_get_char, 512
 - mt_get_length, 512
 - mt_get_null, 513
 - mt_get_number, 513
 - mt_get_type, 513
 - mt_is_empty, 514
 - mt_is_number, 514
 - mt_set_type, 514
- multi_t, 111
 - binary, 111
 - bl, 111
 - dbl, 111
 - fl, 111
 - i16, 112
 - i32, 112
 - i64, 112
 - i8, 112
 - ns, 112
 - st, 112
 - type, 112
 - u16, 112

- u32, [113](#)
- u64, [113](#)
- u8, [113](#)
- val, [113](#)
- multipart_get_boundary
 - servers/http/http.h, [871](#)
 - servers/http/parse.c, [2087](#)
- murmur.c
 - hash_murmur32, [200](#)
 - hash_murmur64, [200](#)
- mutex.c
 - mutex_destroy, [587](#)
 - mutex_init, [587](#)
 - mutex_lock, [588](#)
 - mutex_unlock, [588](#)
- mutex_destroy
 - mutex.c, [587](#)
 - thread.h, [600](#)
- mutex_init
 - mutex.c, [587](#)
 - thread.h, [600](#)
- mutex_lock
 - mutex.c, [588](#)
 - thread.h, [601](#)
- mutex_unlock
 - mutex.c, [588](#)
 - thread.h, [601](#)
- mx_record_t, [114](#)
 - name, [114](#)
 - pref, [114](#)
- my_once_free_d
 - symbols.c, [1999](#)
 - symbols.h, [2047](#)
- mysql.c
 - dash, [1470](#)
 - lib_load_mysql, [1467](#)
 - lib_version, [1470](#)
 - lib_version_mysql, [1467](#)
 - serv_charset, [1471](#)
 - serv_charset_mysql, [1467](#)
 - serv_schema, [1471](#)
 - serv_schema_mysql, [1467](#)
 - serv_type_mysql, [1467](#)
 - serv_version, [1471](#)
 - serv_version_mysql, [1468](#)
 - sql, [1471](#)
 - sql_errno, [1468](#)
 - sql_error, [1468](#)
 - sql_open, [1468](#)
 - sql_ping, [1469](#)
 - sql_pool, [1471](#)
 - sql_start, [1469](#)
 - sql_stop, [1470](#)
 - sql_thread_start, [1470](#)
 - sql_thread_stop, [1470](#)
 - type_embed, [1471](#)
 - type_serv, [1471](#)
 - mysql_affected_rows_d
 - symbols.c, [1999](#)
 - symbols.h, [2047](#)
 - mysql_character_set_name_d
 - symbols.c, [1999](#)
 - symbols.h, [2047](#)
 - mysql_close_d
 - symbols.c, [1999](#)
 - symbols.h, [2047](#)
 - mysql_embedded_d
 - symbols.c, [1999](#)
 - symbols.h, [2047](#)
 - mysql_errno_d
 - symbols.c, [1999](#)
 - symbols.h, [2047](#)
 - mysql_error_d
 - symbols.c, [1999](#)
 - symbols.h, [2047](#)
 - mysql_escape_string_d
 - symbols.c, [1999](#)
 - symbols.h, [2047](#)
 - mysql_fetch_field_d
 - symbols.c, [2000](#)
 - symbols.h, [2047](#)
 - mysql_fetch_row_d
 - symbols.c, [2000](#)
 - symbols.h, [2047](#)
 - mysql_free_result_d
 - symbols.c, [2000](#)
 - symbols.h, [2047](#)
 - mysql_get_client_version_d
 - symbols.c, [2000](#)
 - symbols.h, [2047](#)
 - mysql_get_server_info_d
 - symbols.c, [2000](#)
 - symbols.h, [2048](#)
 - mysql_init_d
 - symbols.c, [2000](#)
 - symbols.h, [2048](#)
 - mysql_insert_id_d
 - symbols.c, [2000](#)
 - symbols.h, [2048](#)
 - mysql_num_fields_d
 - symbols.c, [2000](#)
 - symbols.h, [2048](#)
 - mysql_num_rows_d
 - symbols.c, [2000](#)
 - symbols.h, [2048](#)
 - mysql_options_d
 - symbols.c, [2000](#)
 - symbols.h, [2048](#)
 - mysql_ping_d
 - symbols.c, [2000](#)
 - symbols.h, [2048](#)
 - mysql_real_connect_d
 - symbols.c, [2000](#)
 - symbols.h, [2048](#)

- mysql_real_query_d
 - symbols.c, 2000
 - symbols.h, 2048
- mysql_server_end_d
 - symbols.c, 2001
 - symbols.h, 2048
- mysql_server_init_d
 - symbols.c, 2001
 - symbols.h, 2048
- mysql_set_character_set_d
 - symbols.c, 2001
 - symbols.h, 2048
- mysql_stmt_affected_rows_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_attr_set_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_bind_param_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_bind_result_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_close_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_errno_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_error_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_execute_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_fetch_d
 - symbols.c, 2001
 - symbols.h, 2049
- mysql_stmt_free_result_d
 - symbols.c, 2002
 - symbols.h, 2049
- mysql_stmt_init_d
 - symbols.c, 2002
 - symbols.h, 2049
- mysql_stmt_insert_id_d
 - symbols.c, 2002
 - symbols.h, 2049
- mysql_stmt_num_rows_d
 - symbols.c, 2002
 - symbols.h, 2049
- mysql_stmt_prepare_d
 - symbols.c, 2002
 - symbols.h, 2050
- mysql_stmt_reset_d
 - symbols.c, 2002
 - symbols.h, 2050
- mysql_stmt_result_metadata_d
 - symbols.c, 2002
 - symbols.h, 2050
- mysql_stmt_store_result_d
 - symbols.c, 2002
 - symbols.h, 2050
- mysql_store_result_d
 - symbols.c, 2002
 - symbols.h, 2050
- mysql_thread_end_d
 - symbols.c, 2002
 - symbols.h, 2050
- mysql_thread_id_d
 - symbols.c, 2002
 - symbols.h, 2050
- mysql_thread_init_d
 - symbols.c, 2002
 - symbols.h, 2050
- mysql_thread_safe_d
 - symbols.c, 2003
 - symbols.h, 2050
- naked_message_get
 - providers/prime/messages/messages.c, 1088
 - providers/prime/messages/messages.h, 1186
- naked_message_set
 - providers/prime/messages/messages.c, 1088
 - providers/prime/messages/messages.h, 1186
- name
 - cache_keys_t, 29
 - cache_t, 30
 - dmime_chunk_key_t, 41
 - http_data_t, 76
 - log_level_t, 85
 - magma_keys_t, 86
 - magma_t, 97
 - media_type_t, 109
 - mx_record_t, 114
 - register_session_t, 123
 - relay_keys_t, 125
 - relay_t, 126
 - server_config_t, 128
 - server_keys_t, 129
 - server_t, 131
 - signet_field_key_t, 134
- name_offset
 - signet_field_t, 136
- name_size
 - signet_field_t, 136
- names
 - engine/status/statistics.c, 805
- net_accept
 - listeners.c, 903
- net_init
 - listeners.c, 903
 - network/network.h, 947
- net_listen

- listeners.c, 903
- network/network.h, 947
- net_set_blocking
 - network/network.h, 947
 - options.c, 952
- net_set_buffer_length
 - network/network.h, 947
 - options.c, 952
- net_set_keepalive
 - network/network.h, 948
 - options.c, 953
- net_set_linger
 - network/network.h, 948
 - options.c, 953
- net_set_nodelay
 - network/network.h, 948
 - options.c, 953
- net_set_reuseable_address
 - network/network.h, 949
 - options.c, 954
- net_set_timeout
 - network/network.h, 949
 - options.c, 954
- net_shutdown
 - listeners.c, 903
 - network/network.h, 949
- net_trigger
 - listeners.c, 904
 - network/network.h, 949
- network
 - server_t, 131
- network.c
 - _connect_host, 1561
 - _connect_timeout, 1561
 - CONNECT_TIMEOUT, 1561
- network/http.h
 - HTTP_DATA_ANY, 850
 - HTTP_DATA_GET, 850
 - HTTP_DATA_HEADER, 850
 - HTTP_DATA_POST, 850
 - HTTP_MERGED, 850
 - HTTP_METHOD_CONNECT, 851
 - HTTP_METHOD_DELETE, 850
 - HTTP_METHOD_GET, 850
 - HTTP_METHOD_HEAD, 850
 - HTTP_METHOD_NONE, 850
 - HTTP_METHOD_OPTIONS, 850
 - HTTP_METHOD_POST, 850
 - HTTP_METHOD_PUT, 850
 - HTTP_METHOD_TRACE, 850
 - HTTP_METHOD_UNSUPPORTED, 851
 - HTTP_PORTAL, 850
 - HTTP_DATA, 850
 - http_method_t, 850
- network/imap.h
 - __attribute__, 872
 - imap_arguments_t, 872
 - imap_session_t, 872
- network/meta.h
 - __attribute__, 907
 - CREDENTIAL_AUTH, 905
 - CREDENTIAL_MAIL, 905
 - META_CHECKPOINT_FOLDERS, 905
 - META_CHECKPOINT_MESSAGES, 905
 - META_CHECKPOINT_USER, 905
 - META_GET_ALIASES, 906
 - META_GET_CONTACTS, 906
 - META_GET_FOLDERS, 906
 - META_GET_KEYS, 906
 - META_GET_MESSAGES, 906
 - META_GET_NONE, 906
 - META_LOCKED, 906
 - META_NEED_LOCK, 906
 - META_PROTOCOL_DMTP, 906
 - META_PROTOCOL_GENERIC, 906
 - META_PROTOCOL_IMAP, 906
 - META_PROTOCOL_JSON, 906
 - META_PROTOCOL_NONE, 906
 - META_PROTOCOL_POP, 906
 - META_PROTOCOL_SMTP, 906
 - META_PROTOCOL_WEB, 906
 - META_PROTOCOLCOL_DMAP, 906
 - META_USER_ADVERTISING, 906
 - META_USER_ENCRYPT_DATA, 906
 - META_USER_NONE, 906
 - META_USER_OVERQUOTA, 906
 - META_USER_TLS, 906
 - meta_alert_t, 907
 - meta_alias_t, 907
 - META_CHECKPOINT, 905
 - meta_folder_t, 907
 - META_GET, 905
 - META_LOCK_STATUS, 906
 - meta_message_t, 907
 - META_PROTOCOL, 906
 - META_USER_FLAGS, 906
 - meta_user_t, 907
- network/network.h
 - __attribute__, 934
 - client_close, 934
 - client_connect, 934
 - client_print, 935
 - client_read, 935
 - client_read_line, 935
 - client_secure, 936
 - client_status, 936
 - client_t, 950
 - client_write, 936
 - command_t, 950
 - con_addr, 937
 - con_addr_octet, 937
 - con_addr_presentation, 937
 - con_addr_reversed, 938
 - con_addr_segment, 938

- con_addr_standard, 938
- con_addr_subnet, 939
- con_addr_word, 939
- con_decrement_refs, 939
- con_destroy, 939
- con_flush, 940
- con_increment_refs, 940
- con_init, 940
- con_init_network_buffer, 941
- con_localhost, 941
- con_print, 941
- con_private, 942
- con_read, 942
- con_read_line, 942
- con_reverse_check, 943
- con_reverse_domain, 943
- con_reverse_enqueue, 944
- con_reverse_lookup, 944
- con_reverse_status, 944
- con_secure, 944
- con_status, 945
- con_write_bl, 945
- con_write_ns, 945
- con_write_pl, 946
- con_write_st, 946
- connection_t, 950
- net_init, 947
- net_listen, 947
- net_set_blocking, 947
- net_set_buffer_length, 947
- net_set_keepalive, 948
- net_set_linger, 948
- net_set_nodelay, 948
- net_set_reuseable_address, 949
- net_set_timeout, 949
- net_shutdown, 949
- net_trigger, 949
- protocol_type, 950
- REVERSE_COMPLETE, 934
- REVERSE_EMPTY, 934
- REVERSE_ERROR, 934
- REVERSE_PENDING, 934
- network/sessions.h
 - __attribute__, 971
 - attachment_t, 971
 - composition_t, 971
 - SESSION_STATE_AUTHENTICATED, 971
 - SESSION_STATE_NEUTRAL, 971
 - SESSION_STATE_TERMINATED, 971
 - session_t, 971
- network/smtp.h
 - SMTP_ACTION_BOUNCE, 981
 - SMTP_ACTION_DELETE, 981
 - SMTP_ACTION_ERROR, 981
 - SMTP_ACTION_MARK, 981
 - SMTP_ACTION_MARK_READ, 981
 - SMTP_ACTION_REJECT, 981
 - SMTP_ACTION_UNDEFINED, 981
 - SMTP_FILTER_ACTION_DELETE, 982
 - SMTP_FILTER_ACTION_LABEL, 982
 - SMTP_FILTER_ACTION_MARK_READ, 982
 - SMTP_FILTER_ACTION_MOVE, 982
 - SMTP_FILTER_ACTION_NONE, 982
 - SMTP_FILTER_LOCATION_BODY, 982
 - SMTP_FILTER_LOCATION_ENTIRE, 982
 - SMTP_FILTER_LOCATION_FIELD, 982
 - SMTP_FILTER_LOCATION_HEADER, 982
 - SMTP_FILTER_TYPE_CONTAINS, 982
 - SMTP_FILTER_TYPE_ENDS, 982
 - SMTP_FILTER_TYPE_EXACT, 982
 - SMTP_FILTER_TYPE_REGEXP, 982
 - SMTP_FILTER_TYPE_STARTS, 982
 - SMTP_MARK_FILTERED, 982
 - SMTP_MARK_NONE, 981
 - SMTP_MARK_PHISH, 981
 - SMTP_MARK_RBL, 981
 - SMTP_MARK_READ, 981
 - SMTP_MARK_SPAM, 981
 - SMTP_MARK_SPOOF, 982
 - SMTP_MARK_VIRUS, 981
 - SMTP_OUTCOME_BOUNCE_DKIM, 982
 - SMTP_OUTCOME_BOUNCE_PHISH, 982
 - SMTP_OUTCOME_BOUNCE_RBL, 982
 - SMTP_OUTCOME_BOUNCE_SPAM, 982
 - SMTP_OUTCOME_BOUNCE_SPF, 982
 - SMTP_OUTCOME_BOUNCE_VIRUS, 982
 - SMTP_OUTCOME_PERM_FAILURE, 982
 - SMTP_OUTCOME_SUCESS, 982
 - SMTP_OUTCOME_TEMP_LOCKED, 982
 - SMTP_OUTCOME_TEMP_OVERQUOTA, 982
 - SMTP_OUTCOME_TEMP_SERVER, 982
- network_buffer
 - magma_t, 97
- network_pub.c
 - connect_host, 1563
- next
 - cached_object, 32
 - http_content_t, 75
 - imap_fetch_response_t, 81
 - object_chunk, 117
 - queue_t, 121
 - server_config_t, 128
 - signet_field_t, 136
 - smtp_inbound_prefs_t, 143
 - smtp_recipients_t, 151
- nonce
 - auth_t, 26
- NONE
 - prime.h, 1904
- norm
 - cache_keys_t, 29
 - magma_keys_t, 86
 - relay_keys_t, 125
 - server_keys_t, 129

- normal
 - imap_fetch_dataitems_t, 79
- normal_partial
 - imap_fetch_dataitems_t, 79
- normal_sd
 - server_config_t, 128
- ns
 - multi_t, 112
- NS_ALG_RSASHA1
 - dns.h, 1749
- NS_ALG_RSASHA256
 - dns.h, 1749
- NS_ALG_RSASHA512
 - dns.h, 1749
- ns_alloc
 - nuller.c, 516
 - strings.h, 548
- ns_append
 - nuller.c, 517
 - strings.h, 548
- ns_cleanup
 - strings.h, 540
- ns_cleanup_variadic
 - nuller.c, 517
 - strings.h, 548
- ns_dupe
 - nuller.c, 517
 - strings.h, 549
- ns_empty
 - nuller.c, 518
 - strings.h, 549
- ns_empty_out
 - nuller.c, 518
 - strings.h, 549
- ns_free
 - nuller.c, 518
 - strings.h, 550
- ns_import
 - nuller.c, 519
 - strings.h, 550
- ns_length_get
 - nuller.c, 519
 - strings.h, 550
- ns_length_int
 - nuller.c, 519
 - strings.h, 551
- ns_populated
 - strings.h, 540
- ns_populated_variadic
 - nuller.c, 520
 - strings.h, 551
- ns_wipe
 - nuller.c, 520
 - strings.h, 551
- NULLER
 - strings.h, 540
- nuller.c
 - ns_alloc, 516
 - ns_append, 517
 - ns_cleanup_variadic, 517
 - ns_dupe, 517
 - ns_empty, 518
 - ns_empty_out, 518
 - ns_free, 518
 - ns_import, 519
 - ns_length_get, 519
 - ns_length_int, 519
 - ns_populated_variadic, 520
 - ns_wipe, 520
- NULLER_T
 - strings.h, 543
- nuller_t
 - strings.h, 578
- num_recipients
 - smtp_session_t, 153
- number
 - magma_t, 97
 - objects/sessions/sessions.c, 1236
- numbers.c
 - double_conv, 409
 - float_conv, 409
 - int16_conv_bl, 409
 - int16_conv_ns, 410
 - int16_conv_st, 410
 - int32_conv_bl, 410
 - int32_conv_ns, 411
 - int32_conv_st, 411
 - int64_conv_bl, 411
 - int64_conv_ns, 412
 - int64_conv_st, 412
 - int8_conv_bl, 413
 - int8_conv_ns, 413
 - int8_conv_st, 413
 - size_conv_bl, 414
 - ssize_conv_bl, 414
 - uint16_conv_bl, 414
 - uint16_conv_ns, 415
 - uint16_conv_st, 415
 - uint16_get_no, 416
 - uint16_put_no, 416
 - uint24_get_no, 416
 - uint24_put_no, 416
 - uint32_conv_bl, 417
 - uint32_conv_ns, 417
 - uint32_conv_st, 417
 - uint32_get_no, 418
 - uint32_put_no, 418
 - uint64_conv_bl, 418
 - uint64_conv_ns, 419
 - uint64_conv_pl, 419
 - uint64_conv_st, 420
 - uint8_conv_bl, 420
 - uint8_conv_ns, 420
 - uint8_conv_st, 421

- numbers.h
 - double_conv, 425
 - float_conv, 425
 - int16_clamp, 425
 - int16_conv_bl, 425
 - int16_conv_ns, 426
 - int16_conv_st, 426
 - int16_digits, 427
 - int32_clamp, 427
 - int32_conv_bl, 427
 - int32_conv_ns, 427
 - int32_conv_st, 428
 - int32_digits, 428
 - int64_clamp, 428
 - int64_conv_bl, 429
 - int64_conv_ns, 429
 - int64_conv_st, 429
 - int64_digits, 430
 - int8_clamp, 430
 - int8_conv_bl, 430
 - int8_conv_ns, 431
 - int8_conv_st, 431
 - int8_digits, 432
 - size_conv_bl, 432
 - ssize_conv_bl, 432
 - uint16_clamp, 432
 - uint16_conv_bl, 433
 - uint16_conv_ns, 433
 - uint16_conv_st, 433
 - uint16_digits, 434
 - uint16_get_no, 434
 - uint16_put_no, 434
 - uint24_get_no, 435
 - uint24_put_no, 435
 - uint32_clamp, 435
 - uint32_conv_bl, 435
 - uint32_conv_ns, 436
 - uint32_conv_st, 436
 - uint32_digits, 437
 - uint32_get_no, 437
 - uint32_put_no, 437
 - uint64_clamp, 437
 - uint64_conv_bl, 438
 - uint64_conv_ns, 438
 - uint64_conv_pl, 438
 - uint64_conv_st, 439
 - uint64_digits, 439
 - uint8_clamp, 439
 - uint8_conv_bl, 440
 - uint8_conv_ns, 440
 - uint8_conv_st, 440
 - uint8_digits, 441
- nvp.c
 - nvp_alloc, 396
 - nvp_free, 396
 - nvp_parse, 396
- nvp_alloc
 - formats.h, 394
 - nvp.c, 396
- nvp_free
 - formats.h, 394
 - nvp.c, 396
- nvp_init
 - formats.h, 394
- nvp_parse
 - formats.h, 395
 - nvp.c, 396
- nvp_t, 115
 - __attribute__, 115
 - options, 115
 - pairs, 115
 - tokens, 115
- o2i_ECPublicKey_d
 - symbols.h, 2050
- obj_cache_prune
 - objects.h, 1226
 - objects/objects.c, 1157
- obj_cache_start
 - objects.h, 1226
 - objects/objects.c, 1157
- obj_cache_stop
 - objects.h, 1226
 - objects/objects.c, 1157
- OBJ_cleanup_d
 - symbols.c, 2003
 - symbols.h, 2050
- OBJ_NAME_cleanup_d
 - symbols.c, 2003
 - symbols.h, 2050
- OBJ_nid2sn_d
 - symbols.c, 2003
 - symbols.h, 2051
- OBJECT_ALIASES
 - objects.h, 1225
- OBJECT_CONFIG
 - objects.h, 1225
- OBJECT_CONTACTS
 - objects.h, 1225
- OBJECT_FOLDERS
 - objects.h, 1225
- OBJECT_MESSAGES
 - objects.h, 1225
- OBJECT_USER
 - objects.h, 1225
- object_cache_t, 116
 - meta, 116
 - sessions, 116
- object_chunk, 117
 - data, 117
 - data_size, 117
 - flags, 117
 - next, 117
 - type, 117

- objects
 - objects.h, 1228
 - objects/objects.c, 1158
- objects.h
 - lock_get, 1225
 - lock_release, 1225
 - obj_cache_prune, 1226
 - obj_cache_start, 1226
 - obj_cache_stop, 1226
 - OBJECT_ALIASES, 1225
 - OBJECT_CONFIG, 1225
 - OBJECT_CONTACTS, 1225
 - OBJECT_FOLDERS, 1225
 - OBJECT_MESSAGES, 1225
 - OBJECT_USER, 1225
 - objects, 1228
 - serial_get, 1227
 - serial_increment, 1227
 - serial_reset, 1227
 - user_lock, 1228
 - user_unlock, 1228
- objects/auth/datatier.c
 - auth_data_fetch, 683
 - auth_data_update_legacy, 683
 - auth_data_update_lock, 684
- objects/config/config.c
 - user_config_alloc, 1029
 - user_config_create, 1029
 - user_config_edit, 1030
 - user_config_entry_alloc, 1030
 - user_config_entry_free, 1030
 - user_config_free, 1031
 - user_config_update, 1031
- objects/config/config.h
 - __attribute__, 678
 - USER_CONF_STATUS_CRITICAL, 677
 - USER_CONF_STATUS_NONE, 677
 - user_config_alloc, 678
 - user_config_create, 678
 - user_config_delete, 678
 - user_config_edit, 679
 - user_config_entry_alloc, 679
 - user_config_entry_free, 679
 - user_config_entry_t, 681
 - user_config_fetch, 680
 - user_config_free, 680
 - user_config_t, 681
 - user_config_update, 680
 - user_config_upsert, 681
- objects/config/datatier.c
 - user_config_delete, 685
 - user_config_fetch, 685
 - user_config_upsert, 685
- objects/contacts/contacts.c
 - contact_alloc, 1034
 - contact_create, 1035
 - contact_detail_alloc, 1035
 - contact_detail_free, 1035
 - contact_edit, 1036
 - contact_free, 1036
 - contact_move, 1036
 - contact_name, 1037
 - contact_remove, 1037
 - contact_validate_detail, 1038
 - contact_validate_name, 1038
 - contacts_update, 1038
- objects/contacts/datatier.c
 - contact_delete, 687
 - contact_detail_delete, 687
 - contact_detail_upsert, 688
 - contact_details_fetch, 688
 - contact_insert, 688
 - contact_update, 689
 - contact_update_stamp, 689
 - contacts_fetch, 689
- objects/folders/contacts.c
 - contact_folder_alloc, 1040
 - contact_folder_create, 1040
 - contact_folder_free, 1041
 - contact_folder_remove, 1041
 - contact_folder_rename, 1042
- objects/folders/datatier.c
 - magma_folder_delete, 691
 - magma_folder_fetch, 691
 - magma_folder_insert, 692
 - magma_folder_rename, 692
- objects/folders/folders.c
 - magma_folder_alloc, 1059
 - magma_folder_children, 1059
 - magma_folder_free, 1060
 - magma_folder_funcs, 1060
 - magma_folder_name, 1060
- objects/folders/messages.c
 - message_folder_alloc, 1084
 - message_folder_create, 1084
 - message_folder_free, 1084
 - message_folder_remove, 1085
- objects/mail/cache.c
 - mail_cache_destroy, 615
 - mail_cache_get, 615
 - mail_cache_reset, 616
 - mail_cache_set, 616
 - mail_cache_start, 616
 - mail_cache_stop, 616
 - mail_cache_thread_stop, 617
- objects/mail/counters.c
 - mail_count_received, 1099
 - mail_header_end, 1099
- objects/mail/datatier.c
 - mail_db_delete_message, 693
 - mail_db_hide_message, 693
 - mail_db_insert_duplicate_message, 694
 - mail_db_insert_message, 694
 - mail_db_update_message_folder, 695

- objects/mail/headers.c
 - mail_add_forward_headers, 1101
 - mail_add_inbound_headers, 1102
 - mail_add_outbound_headers, 1102
 - mail_add_required_headers, 1103
 - mail_header_fetch_all, 1103
 - mail_header_fetch_cleaned, 1103
 - mail_header_pop, 1104
 - mail_headers, 1104
 - mail_mod_subject, 1104
 - mail_store_header, 1105
- objects/mail/objects.c
 - mail_create_message, 1155
 - mail_destroy, 1155
 - mail_destroy_message, 1156
 - mail_message, 1156
- objects/messages/datatier.c
 - meta_data_fetch_folder_messages, 696
 - meta_data_fetch_message_tags, 696
 - meta_data_fetch_messages, 697
- objects/messages/messages.c
 - message_alloc, 1086
 - message_free, 1086
 - messages_update, 1086
- objects/messages/messages.h
 - __attribute__, 1179
 - FMESAGE_MAGIC_1, 1176
 - FMESAGE_MAGIC_2, 1176
 - FMESAGE_OPT_COMPRESSED, 1176
 - FMESAGE_OPT_ENCRYPTED, 1177
 - MAIL_MARK_BLACKHOLED, 1179
 - MAIL_MARK_INFECTED, 1179
 - MAIL_MARK_JUNK, 1179
 - MAIL_MARK_PHISHING, 1179
 - MAIL_MARK_SPOOFED, 1179
 - MAIL_STATUS_ANSWERED, 1179
 - MAIL_STATUS_APPENDED, 1179
 - MAIL_STATUS_DELETED, 1179
 - MAIL_STATUS_DRAFT, 1179
 - MAIL_STATUS_EMPTY, 1179
 - MAIL_STATUS_ENCRYPTED, 1179
 - MAIL_STATUS_FLAGGED, 1179
 - MAIL_STATUS_HIDDEN, 1179
 - MAIL_STATUS_NONE, 1179
 - MAIL_STATUS_RECENT, 1179
 - MAIL_STATUS_SECURE, 1179
 - MAIL_STATUS_SEEN, 1179
 - MAIL_STATUS_TAGGED, 1179
 - MAIL_STATUS_ALL_FLAGS, 1177
 - MAIL_STATUS_SYSTEM_FLAGS, 1177
 - MAIL_STATUS_USER_FLAGS, 1177
 - message_alloc, 1179
 - MESSAGE_ENCODING_7BIT, 1177
 - MESSAGE_ENCODING_8BIT, 1177
 - MESSAGE_ENCODING_BASE64, 1177
 - MESSAGE_ENCODING_QUOTED_PRINTABLE, 1177
 - MESSAGE_ENCODING_UNKNOWN, 1178
 - message_free, 1180
 - message_header_t, 1185
 - message_t, 1185
 - MESSAGE_TYPE_HTML, 1178
 - MESSAGE_TYPE_MULTI_ALTERNATIVE, 1178
 - MESSAGE_TYPE_MULTI_MIXED, 1178
 - MESSAGE_TYPE_MULTI_RELATED, 1178
 - MESSAGE_TYPE_MULTI_RFC822, 1178
 - MESSAGE_TYPE_MULTI_UNKOWN, 1178
 - MESSAGE_TYPE_PLAIN, 1178
 - MESSAGE_TYPE_UNKNOWN, 1178
 - messages_update, 1180
 - meta_data_fetch_folder_messages, 1180
 - meta_data_fetch_message_tags, 1181
 - meta_data_fetch_messages, 1181
 - meta_message_by_number, 1181
 - meta_message_dupe, 1182
 - meta_message_free, 1182
 - meta_messages_copier, 1182
 - meta_messages_login_update, 1183
 - meta_messages_mover, 1183
 - meta_messages_update, 1184
 - meta_messages_update_sequences, 1184
- objects/meta/crypto.c
 - meta_crypto_keys_create, 1196
- objects/meta/datatier.c
 - meta_data_acknowledge_alert, 699
 - meta_data_delete_folder, 699
 - meta_data_delete_tag, 699
 - meta_data_fetch_alerts, 700
 - meta_data_fetch_all_tags, 700
 - meta_data_fetch_folders, 700
 - meta_data_fetch_keys, 701
 - meta_data_fetch_mailbox_aliases, 701
 - meta_data_fetch_shard, 701
 - meta_data_fetch_user, 702
 - meta_data_flags_add, 702
 - meta_data_flags_remove, 703
 - meta_data_flags_replace, 703
 - meta_data_insert_folder, 703
 - meta_data_insert_keys, 704
 - meta_data_insert_shard, 704
 - meta_data_insert_tag, 705
 - meta_data_truncate_tags, 705
 - meta_data_update_folder_name, 705
 - meta_data_update_log, 706
- objects/meta/folders.c
 - meta_folder_stats_tag_alloc, 1062
 - meta_folders_by_name, 1062
 - meta_folders_by_number, 1063
 - meta_folders_children, 1063
 - meta_folders_name, 1063
 - meta_folders_stats_tags, 1064
- objects/meta/meta.h
 - alert_alloc, 911
 - alias_alloc, 911
 - meta_alloc, 911

- meta_crypto_keys_create, 912
- meta_data_acknowledge_alert, 912
- meta_data_delete_folder, 912
- meta_data_delete_tag, 913
- meta_data_fetch_alerts, 913
- meta_data_fetch_all_tags, 913
- meta_data_fetch_folders, 914
- meta_data_fetch_keys, 914
- meta_data_fetch_mailbox_aliases, 914
- meta_data_fetch_shard, 915
- meta_data_fetch_user, 915
- meta_data_flags_add, 916
- meta_data_flags_remove, 916
- meta_data_flags_replace, 916
- meta_data_insert_folder, 917
- meta_data_insert_keys, 917
- meta_data_insert_shard, 917
- meta_data_insert_tag, 918
- meta_data_truncate_tags, 918
- meta_data_update_folder_name, 918
- meta_data_update_lock, 919
- meta_data_update_log, 919
- meta_folder_stats_tag_alloc, 919
- meta_folders_by_name, 920
- meta_folders_by_number, 920
- meta_folders_children, 920
- meta_folders_name, 921
- meta_folders_stats_tags, 921
- meta_free, 921
- meta_get, 922
- meta_inx_find, 922
- meta_inx_remove, 923
- meta_update_aliases, 923
- meta_update_contacts, 923
- meta_update_folders, 924
- meta_update_keys, 924
- meta_update_message_folders, 924
- meta_update_realms, 925
- meta_update_user, 925
- meta_user_ref_add, 926
- meta_user_ref_dec, 926
- meta_user_ref_protocol_total, 927
- meta_user_ref_stamp, 927
- meta_user_ref_total, 927
- meta_user_rlock, 927
- meta_user_serial_check, 928
- meta_user_serial_get, 928
- meta_user_serial_set, 928
- meta_user_unlock, 929
- meta_user_wlock, 929
- objects/objects.c
 - obj_cache_prune, 1157
 - obj_cache_start, 1157
 - obj_cache_stop, 1157
 - objects, 1158
- objects/sessions/sessions.c
 - lock, 1236
 - number, 1236
 - sess_create, 1230
 - sess_destroy, 1230
 - sess_get, 1231
 - sess_key, 1231
 - sess_number, 1231
 - sess_ref_add, 1232
 - sess_ref_dec, 1232
 - sess_ref_stamp, 1232
 - sess_ref_total, 1232
 - sess_refresh_check, 1233
 - sess_refresh_flush, 1233
 - sess_refresh_stamp, 1233
 - sess_release, 1234
 - sess_release_attachment, 1234
 - sess_release_composition, 1234
 - sess_serial_check, 1235
 - sess_token, 1235
 - sess_trigger, 1235
 - sess_update, 1236
 - sessions, 1236
- objects/sessions/sessions.h
 - sess_create, 974
 - sess_destroy, 974
 - sess_get, 974
 - sess_key, 975
 - sess_number, 975
 - sess_ref_add, 975
 - sess_ref_dec, 976
 - sess_ref_stamp, 976
 - sess_ref_total, 976
 - sess_refresh_check, 977
 - sess_refresh_flush, 977
 - sess_refresh_stamp, 977
 - sess_release, 977
 - sess_release_attachment, 978
 - sess_release_composition, 978
 - sess_serial_check, 978
 - sess_token, 979
 - sess_trigger, 979
 - sess_update, 979
- objects/warehouse/datatier.c
 - warehouse_fetch_domains, 707
 - warehouse_fetch_patterns, 707
- OCSP_basic_verify_d
 - symbols.h, 2051
- OCSP_BASICRESP_free_d
 - symbols.c, 2003
 - symbols.h, 2051
- OCSP_cert_to_id_d
 - symbols.h, 2051
- OCSP_check_nonce_d
 - symbols.c, 2003
 - symbols.h, 2051
- OCSP_check_validity_d
 - symbols.h, 2051
- OCSP_parse_url_d

- symbols.h, 2051
- OCSP_REQ_CTX_add1_header_d
 - symbols.h, 2051
- OCSP_REQ_CTX_set1_req_d
 - symbols.c, 2003
 - symbols.h, 2051
- OCSP_request_add0_id_d
 - symbols.c, 2003
 - symbols.h, 2051
- OCSP_request_add1_nonce_d
 - symbols.h, 2051
- OCSP_REQUEST_free_d
 - symbols.c, 2003
 - symbols.h, 2051
- OCSP_REQUEST_new_d
 - symbols.c, 2003
 - symbols.h, 2052
- OCSP_REQUEST_print_d
 - symbols.h, 2052
- OCSP_resp_find_status_d
 - symbols.h, 2052
- OCSP_RESPONSE_free_d
 - symbols.c, 2003
 - symbols.h, 2052
- OCSP_response_get1_basic_d
 - symbols.c, 2003
 - symbols.h, 2052
- OCSP_RESPONSE_print_d
 - symbols.h, 2052
- OCSP_response_status_d
 - symbols.c, 2004
 - symbols.h, 2052
- OCSP_response_status_str_d
 - symbols.c, 2004
 - symbols.h, 2052
- OCSP_sendreq_nbio_d
 - symbols.h, 2052
- OCSP_sendreq_new_d
 - symbols.h, 2052
- octet_t
 - host.h, 287
- offset
 - cache_keys_t, 29
 - relay_keys_t, 125
 - server_keys_t, 129
- openssl.c
 - dh2048, 1412
 - dh4096, 1412
 - dhparam_lock, 1412
 - ecdh1024, 1412
 - ecdh512, 1412
 - lib_load_openssl, 1409
 - lib_version_openssl, 1409
 - ssl_ecdh_exchange_callback, 1409
 - ssl_error_string, 1409
 - ssl_locking_callback, 1410
 - ssl_locks, 1412
 - ssl_start, 1410
 - ssl_stop, 1411
 - ssl_thread_id_callback, 1411
 - ssl_thread_stop, 1411
 - ssl_verify_privkey, 1411
 - ssl_version, 1412
- OPENSSL_add_all_algorithms_noconf_d
 - symbols.c, 2004
 - symbols.h, 2052
- OPENSSL_free_d
 - cryptography/cryptography.h, 1349
- options
 - nvp_t, 115
- options.c
 - net_set_blocking, 952
 - net_set_buffer_length, 952
 - net_set_keepalive, 953
 - net_set_linger, 953
 - net_set_nodelay, 953
 - net_set_reuseable_address, 954
 - net_set_timeout, 954
- opts.c
 - st_opt_get, 521
 - st_opt_set, 521
 - st_opt_test, 521
- ORG_ENCRYPTION_KEY
 - prime.h, 1906
- ORG_FULL_SIGNATURE
 - prime.h, 1906
- ORG_IDENTIFIABLE_SIGNATURE
 - prime.h, 1906
- ORG_IDENTIFIER
 - prime.h, 1906
- ORG_PRIMARY_KEY
 - prime.h, 1906
- ORG_SECONDARY_KEY
 - prime.h, 1906
- ORG_SELF_SIGNATURE
 - prime.h, 1906
- org_encrypted_key_get
 - keys/orgs.c, 1863
 - providers/prime/keys/keys.h, 669
- org_encrypted_key_set
 - keys/orgs.c, 1863
 - providers/prime/keys/keys.h, 669
- org_key
 - prime.c, 1899
 - prime.h, 1910
- org_key_alloc
 - keys/orgs.c, 1863
 - providers/prime/keys/keys.h, 669
- org_key_free
 - keys/orgs.c, 1863
 - providers/prime/keys/keys.h, 669
- org_key_generate
 - keys/orgs.c, 1863
 - providers/prime/keys/keys.h, 670

- org_key_get
 - keys/orgs.c, [1864](#)
 - providers/prime/keys/keys.h, [670](#)
- org_key_length
 - keys/orgs.c, [1864](#)
 - providers/prime/keys/keys.h, [670](#)
- org_key_set
 - keys/orgs.c, [1864](#)
 - providers/prime/keys/keys.h, [670](#)
- org_signet
 - prime.c, [1899](#)
 - prime.h, [1910](#)
- org_signet_alloc
 - signets.h, [1915](#)
 - signets/orgs.c, [1865](#)
- org_signet_fingerprint
 - signets.h, [1915](#)
 - signets/orgs.c, [1865](#)
- org_signet_free
 - signets.h, [1916](#)
 - signets/orgs.c, [1865](#)
- org_signet_generate
 - signets.h, [1916](#)
 - signets/orgs.c, [1865](#)
- org_signet_get
 - signets.h, [1916](#)
 - signets/orgs.c, [1866](#)
- org_signet_length
 - signets.h, [1916](#)
 - signets/orgs.c, [1866](#)
- org_signet_set
 - signets.h, [1916](#)
 - signets/orgs.c, [1866](#)
- org_signet_verify
 - signets.h, [1916](#)
 - signets/orgs.c, [1866](#)
- orig_keyslot
 - dmime_chunk_key_t, [41](#)
- origin
 - dmime_message_t, [47](#)
 - dmime_object_t, [49](#)
- origin_display_bounce_sig
 - dmime_message_t, [47](#)
- origin_full_sig
 - dmime_message_t, [47](#)
- origin_meta_bounce_sig
 - dmime_message_t, [47](#)
- OS_NIX
 - ed25519-donna-portable-identify.h, [1655](#)
- other_headers
 - dmime_message_t, [47](#)
 - dmime_object_t, [49](#)
- out_prefs
 - smtp_session_t, [153](#)
- outcome
 - smtp_inbound_prefs_t, [143](#)
- output
 - magma_t, [97](#)
- output.c
 - imap_build_array, [2130](#)
 - imap_build_array_isliteral, [2130](#)
- output_config
 - magma_t, [97](#)
- output_resource_limits
 - magma_t, [97](#)
- overquota
 - smtp_inbound_prefs_t, [144](#)
- overwrite
 - magma_keys_t, [86](#)
- owner
 - allocations.h, [1259](#)
- packed32bignum25519
 - curve25519-donna-sse2.h, [1643](#)
- packed64bignum25519
 - curve25519-donna-sse2.h, [1643](#)
- packedelem32
 - curve25519-donna-sse2.h, [1643](#)
- packedelem32_t, [118](#)
 - u, [118](#)
 - v, [118](#)
- packedelem64
 - curve25519-donna-sse2.h, [1643](#)
- packedelem64_t, [119](#)
 - u, [119](#)
 - v, [119](#)
- packedelem8
 - curve25519-donna-sse2.h, [1643](#)
- packedelem8_t, [120](#)
 - u, [120](#)
 - v, [120](#)
- page_length
 - magma_t, [97](#)
- pages
 - content.c, [2109](#)
 - magma_t, [97](#)
- pairs
 - __attribute__, [19](#)
 - nvp_t, [115](#)
- parameters.c
 - dh2048, [1416](#)
 - dh4096, [1416](#)
 - dh_exchange_2048, [1414](#)
 - dh_exchange_4096, [1414](#)
 - dh_params_2048, [1415](#)
 - dh_params_4096, [1415](#)
 - dh_params_generate, [1415](#)
 - dh_params_generate_callback, [1415](#)
 - dh_static_2048, [1415](#)
 - dh_static_4096, [1416](#)
 - dhparam_lock, [1416](#)
- params
 - __attribute__, [19](#)
- parse.h

- dime_prsr_envelope_destroy, 1600
- dime_prsr_envelope_format, 1600
- dime_prsr_envelope_parse, 1600
- dime_prsr_headers_create, 1601
- dime_prsr_headers_destroy, 1601
- dime_prsr_headers_format, 1601
- dime_prsr_headers_parse, 1601
- dmime_header_keys, 1602
- dmime_header_type_t, 1599
- HEADER_TYPE_CC, 1599
- HEADER_TYPE_DATE, 1599
- HEADER_TYPE_FROM, 1600
- HEADER_TYPE_NONE, 1600
- HEADER_TYPE_ORGANIZATION, 1600
- HEADER_TYPE_SUBJECT, 1600
- HEADER_TYPE_TO, 1599
- parse_address.c
 - imap_parse_address, 2131
 - imap_parse_address_breaker, 2131
 - imap_parse_address_part, 2131
 - imap_parse_address_put, 2131
- parse_dime_record
 - mrec_pub.c, 1768
- parser.c
 - dime_prsr_envelope_destroy, 1603
 - dime_prsr_envelope_format, 1603
 - dime_prsr_envelope_parse, 1604
 - dime_prsr_headers_create, 1604
 - dime_prsr_headers_destroy, 1604
 - dime_prsr_headers_format, 1604
 - dime_prsr_headers_parse, 1605
 - dmime_header_keys, 1605
- parsers/bitwise.c
 - st_and, 367
 - st_bitwise, 367
 - st_not, 368
 - st_or, 368
 - st_xor, 368
- parsing.c
 - mail_domain_get, 1161
 - mail_extract_address, 1161
- part_buffer
 - parts.c, 1893
 - parts.h, 1894
- part_decrypt
 - parts.c, 1893
 - parts.h, 1894
- part_encrypt
 - parts.c, 1893
 - parts.h, 1894
- parts.c
 - part_buffer, 1893
 - part_decrypt, 1893
 - part_encrypt, 1893
- parts.h
 - part_buffer, 1894
 - part_decrypt, 1894
- part_encrypt, 1894
- password
 - auth_stacie_t, 23
 - magma_t, 97
 - register_session_t, 123
 - teacher_data_t, 157
- passwords.c
 - stacie_derive_key, 1942
 - stacie_derive_rounds, 1943
 - stacie_derive_seed, 1943
- path
 - magma_t, 97
- paths.c
 - mail_create_directory, 1162
 - mail_message_path, 1162
 - mail_path_finder, 1162
- pattern_check
 - patterns.c, 1248
 - warehouse.h, 1255
- pattern_start
 - patterns.c, 1248
 - warehouse.h, 1255
- pattern_stop
 - patterns.c, 1249
 - warehouse.h, 1256
- pattern_update
 - patterns.c, 1249
 - warehouse.h, 1256
- patterns.c
 - pattern_check, 1248
 - pattern_start, 1248
 - pattern_stop, 1249
 - pattern_update, 1249
 - patterns_list, 1249
 - patterns_mutex, 1249
 - patterns_stamp, 1249
- patterns_list
 - patterns.c, 1249
- patterns_mutex
 - patterns.c, 1249
- patterns_stamp
 - patterns.c, 1249
- payload
 - dmime_chunk_key_t, 41
- PAYLOAD_TYPE_EPHEMERAL
 - dmessage/common.h, 1572
- PAYLOAD_TYPE_NONE
 - dmessage/common.h, 1572
- PAYLOAD_TYPE_SIGNATURE
 - dmessage/common.h, 1572
- PAYLOAD_TYPE_STANDARD
 - dmessage/common.h, 1572
- peek
 - imap_fetch_dataitems_t, 79
- peek_partial
 - imap_fetch_dataitems_t, 79
- pem.c

- prime_pem_begin, [1921](#)
- prime_pem_end, [1922](#)
- prime_pem_endings, [1922](#)
- prime_pem_headings, [1922](#)
- PRIME_PEM_LINE_WRAP_CHARS, [1921](#)
- PRIME_PEM_LINE_WRAP_LENGTH, [1921](#)
- PRIME_PEM_LINE_WRAP_TYPE, [1921](#)
- prime_pem_unwrap, [1922](#)
- prime_pem_wrap, [1922](#)
- perf_rdtsc
 - performance.c, [797](#)
 - status.h, [813](#)
- performance.c
 - perf_rdtsc, [797](#)
- persists
 - cached_object, [32](#)
- phish
 - smtp_inbound_prefs_t, [144](#)
- phishaction
 - smtp_inbound_prefs_t, [144](#)
- pid
 - status.c, [809](#)
- pk
 - generated_data_t, [74](#)
 - test_data_t, [159](#)
- PL_AVAILABLE
 - buckets.h, [169](#)
- PL_ERROR
 - buckets.h, [169](#)
- PL_RESERVED
 - buckets.h, [169](#)
- pl_char_get
 - shortcuts.c, [528](#)
 - strings.h, [552](#)
- pl_clone
 - shortcuts.c, [529](#)
 - strings.h, [552](#)
- pl_data_get
 - shortcuts.c, [529](#)
 - strings.h, [552](#)
- pl_empty
 - shortcuts.c, [529](#)
 - strings.h, [552](#)
- pl_get_embraced
 - core/parsers/parsers.h, [446](#)
 - token.c, [476](#)
- pl_inc
 - shortcuts.c, [529](#)
 - strings.h, [553](#)
- pl_init
 - shortcuts.c, [530](#)
 - strings.h, [553](#)
- pl_length_get
 - shortcuts.c, [530](#)
 - strings.h, [554](#)
- pl_length_int
 - shortcuts.c, [531](#)
- strings.h, [554](#)
- pl_null
 - shortcuts.c, [531](#)
 - strings.h, [554](#)
- pl_set
 - shortcuts.c, [531](#)
 - strings.h, [555](#)
- pl_shrink_before_characters
 - core/parsers/parsers.h, [446](#)
 - token.c, [476](#)
- pl_skip_characters
 - core/parsers/parsers.h, [446](#)
 - token.c, [476](#)
- pl_skip_to_characters
 - core/parsers/parsers.h, [447](#)
 - token.c, [476](#)
- pl_starts_with_char
 - shortcuts.c, [531](#)
 - strings.h, [555](#)
- pl_trim
 - core/parsers/parsers.h, [447](#)
 - trim.c, [482](#)
- pl_trim_end
 - core/parsers/parsers.h, [447](#)
 - trim.c, [482](#)
- pl_trim_start
 - core/parsers/parsers.h, [447](#)
 - trim.c, [482](#)
- pl_update_start
 - core/parsers/parsers.h, [448](#)
 - token.c, [477](#)
- PLACER
 - strings.h, [540](#)
- PLACER_T
 - strings.h, [543](#)
- placer_t
 - strings.h, [578](#)
- plan
 - register_session_t, [123](#)
- PNG
 - signet/common.h, [1576](#)
- png.c
 - lib_load_png, [1838](#)
 - lib_version_png, [1838](#)
- png_access_version_number_d
 - symbols.h, [2052](#)
- points
 - batch_heap_t, [28](#)
- policy
 - dime_record_t, [39](#)
- pool
 - magma_t, [98](#)
- pool.c
 - pool_alloc, [181](#)
 - pool_free, [182](#)
 - pool_get_available, [182](#)
 - pool_get_count, [182](#)

- pool_get_failures, 183
- pool_get_obj, 183
- pool_get_status, 183
- pool_get_timeout, 184
- pool_pull, 184
- pool_release, 184
- pool_set_obj, 185
- pool_set_status, 185
- pool_swap_obj, 186
- pool_alloc
 - buckets.h, 173
 - pool.c, 181
- pool_free
 - buckets.h, 174
 - pool.c, 182
- pool_get_available
 - buckets.h, 174
 - pool.c, 182
- pool_get_count
 - buckets.h, 174
 - pool.c, 182
- pool_get_failures
 - buckets.h, 175
 - pool.c, 183
- pool_get_obj
 - buckets.h, 175
 - pool.c, 183
- pool_get_status
 - buckets.h, 175
 - pool.c, 183
- pool_get_timeout
 - buckets.h, 176
 - pool.c, 184
- pool_pull
 - buckets.h, 176
 - pool.c, 184
- pool_release
 - buckets.h, 176
 - pool.c, 184
- pool_set_obj
 - buckets.h, 177
 - pool.c, 185
- pool_set_status
 - buckets.h, 177
 - pool.c, 185
- pool_swap_obj
 - buckets.h, 178
 - pool.c, 186
- pool_t
 - buckets.h, 179
- POP
 - engine/config/servers/servers.h, 753
- pop.c
 - pop_capa, 2139
 - pop_dele, 2139
 - pop_init, 2139
 - pop_invalid, 2139
 - pop_last, 2140
 - pop_list, 2140
 - pop_noop, 2140
 - pop_pass, 2141
 - pop_quit, 2141
 - pop_retr, 2141
 - pop_rset, 2142
 - pop_starttls, 2142
 - pop_stat, 2142
 - pop_top, 2143
 - pop_uidl, 2143
 - pop_user, 2143
- pop_capa
 - pop.c, 2139
 - servers/pop/pop.h, 957
- pop_commands
 - servers/pop/commands.h, 1635
- pop_compare
 - servers/pop/commands.c, 1619
 - servers/pop/pop.h, 958
- pop_dele
 - pop.c, 2139
 - servers/pop/pop.h, 958
- pop_get_last
 - mailbox.c, 2136
 - servers/pop/pop.h, 958
- pop_get_message
 - mailbox.c, 2136
 - servers/pop/pop.h, 958
- pop_init
 - pop.c, 2139
 - servers/pop/pop.h, 959
- pop_invalid
 - pop.c, 2139
 - servers/pop/pop.h, 959
- pop_last
 - pop.c, 2140
 - servers/pop/pop.h, 959
- pop_last_error
 - error.c, 1523
 - error.h, 1535
- pop_list
 - pop.c, 2140
 - servers/pop/pop.h, 960
- pop_noop
 - pop.c, 2140
 - servers/pop/pop.h, 960
- pop_num_parse
 - servers/pop/parse.c, 2094
 - servers/pop/pop.h, 960
- pop_pass
 - pop.c, 2141
 - servers/pop/pop.h, 961
- pop_pass_parse
 - servers/pop/parse.c, 2094
 - servers/pop/pop.h, 961
- pop_process

- servers/pop/commands.c, 1619
- servers/pop/pop.h, 961
- pop_quit
 - pop.c, 2141
 - servers/pop/pop.h, 962
- pop_requeue
 - servers/pop/commands.c, 1619
 - servers/pop/pop.h, 962
- pop_retr
 - pop.c, 2141
 - servers/pop/pop.h, 962
- pop_rset
 - pop.c, 2142
 - servers/pop/pop.h, 962
- pop_session_destroy
 - servers/pop/pop.h, 963
 - servers/pop/sessions.c, 1241
- pop_session_reset
 - servers/pop/pop.h, 963
 - servers/pop/sessions.c, 1241
- pop_sort
 - servers/pop/commands.c, 1619
 - servers/pop/pop.h, 963
- pop_starttls
 - pop.c, 2142
 - servers/pop/pop.h, 963
- pop_stat
 - pop.c, 2142
 - servers/pop/pop.h, 964
- pop_top
 - pop.c, 2143
 - servers/pop/pop.h, 964
- pop_top_parse
 - servers/pop/parse.c, 2095
 - servers/pop/pop.h, 964
- pop_total_messages
 - mailbox.c, 2137
 - servers/pop/pop.h, 965
- pop_total_size
 - mailbox.c, 2137
 - servers/pop/pop.h, 965
- pop_uidl
 - pop.c, 2143
 - servers/pop/pop.h, 965
- pop_user
 - pop.c, 2143
 - servers/pop/pop.h, 966
- pop_user_parse
 - servers/pop/parse.c, 2095
 - servers/pop/pop.h, 966
- port
 - __attribute__, 19
 - cache_t, 30
 - magma_t, 98
 - relay_t, 126
 - server_t, 131
- portal
 - __attribute__, 19
 - magma_t, 98
- portal.c
 - portal_print_login, 2203
 - portal_process, 2203
- portal.h
 - JSON_RPC_2_ERROR_APPLICATION, 2210
 - JSON_RPC_2_ERROR_PARSE_CHAR, 2210
 - JSON_RPC_2_ERROR_PARSE_ENCODING, 2210
 - JSON_RPC_2_ERROR_PARSE_MALFORMED, 2210
 - JSON_RPC_2_ERROR_SERVER_INTERNAL, 2210
 - JSON_RPC_2_ERROR_SERVER_METHOD_PARAMS, 2210
 - JSON_RPC_2_ERROR_SERVER_METHOD_UNAVAIL, 2210
 - JSON_RPC_2_ERROR_SERVER_REQUEST, 2210
 - JSON_RPC_2_ERROR_SYSTEM, 2210
 - JSON_RPC_2_ERROR_TRANSPORT, 2210
 - MAGMA_PORTAL_VERSION, 2209
 - PORTAL_ENDPOINT_ACTION_ADD, 2210
 - PORTAL_ENDPOINT_ACTION_INVALID, 2210
 - PORTAL_ENDPOINT_ACTION_LIST, 2210
 - PORTAL_ENDPOINT_ACTION_REMOVE, 2210
 - PORTAL_ENDPOINT_ACTION_REPLACE, 2210
 - PORTAL_ENDPOINT_CONTEXT_CONTACTS, 2210
 - PORTAL_ENDPOINT_CONTEXT_HELP, 2210
 - PORTAL_ENDPOINT_CONTEXT_INVALID, 2210
 - PORTAL_ENDPOINT_CONTEXT_MAIL, 2210
 - PORTAL_ENDPOINT_CONTEXT_SETTINGS, 2210
 - PORTAL_ENDPOINT_ERROR_AD, 2211
 - PORTAL_ENDPOINT_ERROR_ALERT_ACKNOWLEDGE, 2211
 - PORTAL_ENDPOINT_ERROR_ALERT_LIST, 2211
 - PORTAL_ENDPOINT_ERROR_ALIASES, 2211
 - PORTAL_ENDPOINT_ERROR_ATTACHMENTS_ADD, 2211
 - PORTAL_ENDPOINT_ERROR_ATTACHMENTS_PROGRESS, 2211
 - PORTAL_ENDPOINT_ERROR_ATTACHMENTS_REMOVE, 2211
 - PORTAL_ENDPOINT_ERROR_AUTH, 2211
 - PORTAL_ENDPOINT_ERROR_CONFIG_EDIT, 2211
 - PORTAL_ENDPOINT_ERROR_CONFIG_LOAD, 2211
 - PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION, 2211
 - PORTAL_ENDPOINT_ERROR_CONTACTS_ADD, 2211
 - PORTAL_ENDPOINT_ERROR_CONTACTS_COPY, 2211
 - PORTAL_ENDPOINT_ERROR_CONTACTS_EDIT, 2211
 - PORTAL_ENDPOINT_ERROR_CONTACTS_LIST, 2211
 - PORTAL_ENDPOINT_ERROR_CONTACTS_LOAD, 2211
 - PORTAL_ENDPOINT_ERROR_CONTACTS_MOVE, 2211
 - PORTAL_ENDPOINT_ERROR_CONTACTS_REMOVE, 2211
 - PORTAL_ENDPOINT_ERROR_COOKIES, 2211
 - PORTAL_ENDPOINT_ERROR_FOLDERS_ADD, 2211
 - PORTAL_ENDPOINT_ERROR_FOLDERS_LIST, 2211
 - PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE, 2212
 - PORTAL_ENDPOINT_ERROR_FOLDERS_RENAME, 2212

PORTAL_ENDPOINT_ERROR_FOLDERS_TAGS, 2212
PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION, 2211
PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD, 2211
PORTAL_ENDPOINT_ERROR_LOGOUT, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_COMPOSE, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_COPY, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_FLAG, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_LIST, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_LOAD, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_MOVE, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_REMOVE, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_SEND, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_TAG, 2212
PORTAL_ENDPOINT_ERROR_MESSAGES_TAGS, 2212
PORTAL_ENDPOINT_ERROR_META, 2212
PORTAL_ENDPOINT_ERROR_MODE, 2211
PORTAL_ENDPOINT_ERROR_NONE, 2211
PORTAL_ENDPOINT_ERROR_READ, 2211
PORTAL_ENDPOINT_ERROR_REFERENCE, 2211
PORTAL_ENDPOINT_ERROR_SCRAPE, 2211
PORTAL_ENDPOINT_ERROR_SCRAPE_ADD, 2211
PORTAL_ENDPOINT_ERROR_SEARCH, 2212
PORTAL_ENDPOINT_ERROR_SETTINGS_CHANGEPASS, 2212
PORTAL_ENDPOINT_ERROR_SETTINGS_IDENTITY, 2212
PORTAL_ENDPOINT_ERROR_SYSTEM_FLAG, 2211
PORTAL_ENDPOINT_MESSAGE_ATTACHMENTS, 2211
PORTAL_ENDPOINT_MESSAGE_BODY, 2211
PORTAL_ENDPOINT_MESSAGE_HEADER, 2210
PORTAL_ENDPOINT_MESSAGE_INFO, 2211
PORTAL_ENDPOINT_MESSAGE_META, 2210
PORTAL_ENDPOINT_MESSAGE_SECURITY, 2210
PORTAL_ENDPOINT_MESSAGE_SERVER, 2210
PORTAL_ENDPOINT_MESSAGE_SOURCE, 2210
portal_config_collection, 2212
portal_config_entry, 2212
portal_config_entry_flags, 2212
portal_contact_detail_flags, 2213
portal_contact_details, 2213
portal_debug, 2213
portal_endpoint, 2214
portal_endpoint_ad, 2214
portal_endpoint_alert_acknowledge, 2214
portal_endpoint_alert_list, 2215
portal_endpoint_aliases, 2215
portal_endpoint_attachments_add, 2215
portal_endpoint_attachments_progress, 2216
portal_endpoint_attachments_remove, 2216
portal_endpoint_auth, 2216
portal_endpoint_compare, 2216
portal_endpoint_config_edit, 2217
portal_endpoint_config_load, 2217
portal_endpoint_contacts_add, 2217
portal_endpoint_contacts_copy, 2218
portal_endpoint_contacts_edit, 2218
portal_endpoint_contacts_list, 2218
portal_endpoint_contacts_load, 2219
portal_endpoint_contacts_move, 2219
portal_endpoint_contacts_remove, 2219
portal_endpoint_cookies, 2220
portal_endpoint_error, 2220
portal_endpoint_folders_add, 2221
portal_endpoint_folders_list, 2221
portal_endpoint_folders_remove, 2221
portal_endpoint_folders_rename, 2222
portal_endpoint_folders_tags, 2222
portal_endpoint_logout, 2222
portal_endpoint_messages_compose, 2223
portal_endpoint_messages_copy, 2223
portal_endpoint_messages_flag, 2223
portal_endpoint_messages_list, 2224
portal_endpoint_messages_load, 2224
portal_endpoint_messages_move, 2224
portal_endpoint_messages_remove, 2225
portal_endpoint_messages_send, 2225
portal_endpoint_messages_tag, 2225
portal_endpoint_messages_tags, 2226
portal_endpoint_response, 2226
portal_endpoint_scrape, 2227
portal_endpoint_scrape_add, 2227
portal_endpoint_search, 2227
portal_endpoint_sort, 2228
portal_folder_contacts_add, 2228
portal_folder_contacts_remove, 2228
portal_folder_mail_add, 2229
portal_folder_mail_remove, 2229
portal_message_attachments, 2229
portal_message_body, 2230
portal_message_flags_array, 2230
portal_message_header, 2230
portal_message_info, 2231
portal_message_meta, 2231
portal_message_security, 2231
portal_message_server, 2231
portal_message_source, 2231
portal_message_tags_array, 2232
portal_meta, 2232
portal_outbound_checks, 2232
portal_parse_action, 2233
portal_parse_context, 2233
portal_parse_flags, 2233
portal_parse_json_str_array, 2233
portal_parse_sections, 2234
portal_print_login, 2234
portal_process, 2235
portal_settings_identity, 2235
portal_smtp_create_data, 2235
portal_smtp_merge_headers, 2236
portal_smtp_relay_message, 2236
portal_upload, 2237

PORTAL_ENDPOINT_ACTION_ADD
portal.h, 2210

PORTAL_ENDPOINT_ACTION_INVALID
portal.h, 2210

PORTAL_ENDPOINT_ACTION_LIST
portal.h, 2210

PORTAL_ENDPOINT_ACTION_REMOVE
portal.h, 2210

PORTAL_ENDPOINT_ACTION_REPLACE
portal.h, 2210

PORTAL_ENDPOINT_CONTEXT_CONTACTS
portal.h, 2210

PORTAL_ENDPOINT_CONTEXT_HELP
portal.h, 2210

PORTAL_ENDPOINT_CONTEXT_INVALID
portal.h, 2210

PORTAL_ENDPOINT_CONTEXT_MAIL
portal.h, 2210

PORTAL_ENDPOINT_CONTEXT_SETTINGS
portal.h, 2210

PORTAL_ENDPOINT_ERROR_AD
portal.h, 2211

PORTAL_ENDPOINT_ERROR_ALERT_ACKNOWLEDGE
portal.h, 2211

PORTAL_ENDPOINT_ERROR_ALERT_LIST
portal.h, 2211

PORTAL_ENDPOINT_ERROR_ALIASES
portal.h, 2211

PORTAL_ENDPOINT_ERROR_ATTACHMENTS_ADD
portal.h, 2211

PORTAL_ENDPOINT_ERROR_ATTACHMENTS_PROGRESS
portal.h, 2211

PORTAL_ENDPOINT_ERROR_ATTACHMENTS_REMOVE
portal.h, 2211

PORTAL_ENDPOINT_ERROR_AUTH
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONFIG_EDIT
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONFIG_LOAD
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONSTRAINT_VIOLATION
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONTACTS_ADD
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONTACTS_COPY
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONTACTS_EDIT
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONTACTS_LIST
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONTACTS_LOAD
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONTACTS_MOVE
portal.h, 2211

PORTAL_ENDPOINT_ERROR_CONTACTS_REMOVE
portal.h, 2211

PORTAL_ENDPOINT_ERROR_COOKIES
portal.h, 2211

PORTAL_ENDPOINT_ERROR_FOLDERS_ADD
portal.h, 2211

PORTAL_ENDPOINT_ERROR_FOLDERS_LIST
portal.h, 2211

PORTAL_ENDPOINT_ERROR_FOLDERS_REMOVE
portal.h, 2212

PORTAL_ENDPOINT_ERROR_FOLDERS_RENAME
portal.h, 2212

PORTAL_ENDPOINT_ERROR_FOLDERS_TAGS
portal.h, 2212

PORTAL_ENDPOINT_ERROR_ILLEGAL_COMBINATION
portal.h, 2211

PORTAL_ENDPOINT_ERROR_INVALID_KEYWORD
portal.h, 2211

PORTAL_ENDPOINT_ERROR_LOGOUT
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_COMPOSE
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_COPY
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_FLAG
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_LIST
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_LOAD
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_MOVE
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_REMOVE
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_SEND
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_TAG
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MESSAGES_TAGS
portal.h, 2212

PORTAL_ENDPOINT_ERROR_META
portal.h, 2212

PORTAL_ENDPOINT_ERROR_MODE
portal.h, 2211

PORTAL_ENDPOINT_ERROR_NONE
portal.h, 2211

PORTAL_ENDPOINT_ERROR_READ
portal.h, 2211

PORTAL_ENDPOINT_ERROR_REFERENCE
portal.h, 2211

PORTAL_ENDPOINT_ERROR_SCRAPE
portal.h, 2211

PORTAL_ENDPOINT_ERROR_SCRAPE_ADD
portal.h, 2211

PORTAL_ENDPOINT_ERROR_SEARCH
portal.h, 2212

PORTAL_ENDPOINT_ERROR_SETTINGS_CHANGEPASS
portal.h, 2212

PORTAL_ENDPOINT_ERROR_SETTINGS_IDENTITY
portal.h, 2212

PORTAL_ENDPOINT_ERROR_SYSTEM_FLAG
 portal.h, 2211
 PORTAL_ENDPOINT_MESSAGE_ATTACHMENTS
 portal.h, 2211
 PORTAL_ENDPOINT_MESSAGE_BODY
 portal.h, 2211
 PORTAL_ENDPOINT_MESSAGE_HEADER
 portal.h, 2210
 PORTAL_ENDPOINT_MESSAGE_INFO
 portal.h, 2211
 PORTAL_ENDPOINT_MESSAGE_META
 portal.h, 2210
 PORTAL_ENDPOINT_MESSAGE_SECURITY
 portal.h, 2210
 PORTAL_ENDPOINT_MESSAGE_SERVER
 portal.h, 2210
 PORTAL_ENDPOINT_MESSAGE_SOURCE
 portal.h, 2210
 portal_stat_emails_received_today
 statistics.h, 2254
 portal_stat_emails_received_week
 statistics.h, 2254
 portal_stat_emails_sent_today
 statistics.h, 2254
 portal_stat_emails_sent_week
 statistics.h, 2254
 portal_stat_total_users
 statistics.h, 2254
 portal_stat_users_checked_email_today
 statistics.h, 2254
 portal_stat_users_checked_email_week
 statistics.h, 2254
 portal_stat_users_registered_today
 statistics.h, 2254
 portal_stat_users_registered_total
 statistics.h, 2254
 portal_stat_users_registered_week
 statistics.h, 2254
 portal_stat_users_sent_email_today
 statistics.h, 2254
 portal_stat_users_sent_email_week
 statistics.h, 2254
 portal_config_collection
 portal.h, 2212
 web/portal/config.c, 1032
 portal_config_entry
 portal.h, 2212
 web/portal/config.c, 1032
 portal_config_entry_flags
 portal.h, 2212
 web/portal/config.c, 1032
 portal_contact_detail_flags
 portal.h, 2213
 web/portal/contacts.c, 1043
 portal_contact_details
 portal.h, 2213
 web/portal/contacts.c, 1043
 portal_debug
 endpoint.c, 2181
 portal.h, 2213
 portal_endpoint
 endpoint.c, 2181
 portal.h, 2214
 portal_endpoint_ad
 endpoint.c, 2182
 portal.h, 2214
 portal_endpoint_alert_acknowledge
 endpoint.c, 2182
 portal.h, 2214
 portal_endpoint_alert_list
 endpoint.c, 2182
 portal.h, 2215
 portal_endpoint_aliases
 endpoint.c, 2183
 portal.h, 2215
 portal_endpoint_attachments_add
 endpoint.c, 2183
 portal.h, 2215
 portal_endpoint_attachments_progress
 endpoint.c, 2183
 portal.h, 2216
 portal_endpoint_attachments_remove
 endpoint.c, 2183
 portal.h, 2216
 portal_endpoint_auth
 endpoint.c, 2184
 portal.h, 2216
 portal_endpoint_compare
 endpoint.c, 2184
 portal.h, 2216
 portal_endpoint_config_edit
 endpoint.c, 2184
 portal.h, 2217
 portal_endpoint_config_load
 endpoint.c, 2185
 portal.h, 2217
 portal_endpoint_contacts_add
 endpoint.c, 2185
 portal.h, 2217
 portal_endpoint_contacts_copy
 endpoint.c, 2186
 portal.h, 2218
 portal_endpoint_contacts_edit
 endpoint.c, 2186
 portal.h, 2218
 portal_endpoint_contacts_list
 endpoint.c, 2186
 portal.h, 2218
 portal_endpoint_contacts_load
 endpoint.c, 2187
 portal.h, 2219
 portal_endpoint_contacts_move
 endpoint.c, 2187
 portal.h, 2219

portal_endpoint_contacts_remove
 endpoint.c, [2187](#)
 portal.h, [2219](#)

portal_endpoint_cookies
 endpoint.c, [2188](#)
 portal.h, [2220](#)

portal_endpoint_error
 endpoint.c, [2188](#)
 portal.h, [2220](#)

portal_endpoint_folders_add
 endpoint.c, [2188](#)
 portal.h, [2221](#)

portal_endpoint_folders_list
 endpoint.c, [2189](#)
 portal.h, [2221](#)

portal_endpoint_folders_remove
 endpoint.c, [2189](#)
 portal.h, [2221](#)

portal_endpoint_folders_rename
 endpoint.c, [2190](#)
 portal.h, [2222](#)

portal_endpoint_folders_tags
 endpoint.c, [2190](#)
 portal.h, [2222](#)

portal_endpoint_logout
 endpoint.c, [2190](#)
 portal.h, [2222](#)

portal_endpoint_messages_compose
 endpoint.c, [2191](#)
 portal.h, [2223](#)

portal_endpoint_messages_copy
 endpoint.c, [2191](#)
 portal.h, [2223](#)

portal_endpoint_messages_flag
 endpoint.c, [2191](#)
 portal.h, [2223](#)

portal_endpoint_messages_list
 endpoint.c, [2191](#)
 portal.h, [2224](#)

portal_endpoint_messages_load
 endpoint.c, [2192](#)
 portal.h, [2224](#)

portal_endpoint_messages_move
 endpoint.c, [2192](#)
 portal.h, [2224](#)

portal_endpoint_messages_remove
 endpoint.c, [2192](#)
 portal.h, [2225](#)

portal_endpoint_messages_send
 endpoint.c, [2193](#)
 portal.h, [2225](#)

portal_endpoint_messages_tag
 endpoint.c, [2193](#)
 portal.h, [2225](#)

portal_endpoint_messages_tags
 endpoint.c, [2194](#)
 portal.h, [2226](#)

portal_endpoint_response
 endpoint.c, [2194](#)
 portal.h, [2226](#)

portal_endpoint_scrape
 endpoint.c, [2195](#)
 portal.h, [2227](#)

portal_endpoint_scrape_add
 endpoint.c, [2195](#)
 portal.h, [2227](#)

portal_endpoint_search
 endpoint.c, [2195](#)
 portal.h, [2227](#)

portal_endpoint_sort
 endpoint.c, [2195](#)
 portal.h, [2228](#)

portal_folder_contacts_add
 portal.h, [2228](#)
 web/portal/folders.c, [1070](#)

portal_folder_contacts_remove
 portal.h, [2228](#)
 web/portal/folders.c, [1070](#)

portal_folder_mail_add
 portal.h, [2229](#)
 web/portal/folders.c, [1071](#)

portal_folder_mail_remove
 portal.h, [2229](#)
 web/portal/folders.c, [1071](#)

portal_get_upload_attachment
 endpoint.c, [2196](#)

portal_message_attachments
 portal.h, [2229](#)
 web/portal/messages.c, [1091](#)

portal_message_body
 portal.h, [2230](#)
 web/portal/messages.c, [1091](#)

portal_message_flags_array
 portal.h, [2230](#)
 web/portal/flags.c, [2122](#)

portal_message_header
 portal.h, [2230](#)
 web/portal/messages.c, [1091](#)

portal_message_info
 portal.h, [2231](#)
 web/portal/messages.c, [1092](#)

portal_message_meta
 portal.h, [2231](#)
 web/portal/messages.c, [1092](#)

portal_message_security
 portal.h, [2231](#)
 web/portal/messages.c, [1092](#)

portal_message_server
 portal.h, [2231](#)
 web/portal/messages.c, [1092](#)

portal_message_source
 portal.h, [2231](#)
 web/portal/messages.c, [1093](#)

portal_message_tags_array

- portal.h, 2232
- web/portal/flags.c, 2122
- portal_meta
 - endpoint.c, 2196
 - portal.h, 2232
- portal_methods
 - methods.h, 2202
- portal_outbound_checks
 - mail.c, 2199
 - portal.h, 2232
- portal_parse_action
 - portal.h, 2233
 - web/portal/parse.c, 2098
- portal_parse_context
 - portal.h, 2233
 - web/portal/parse.c, 2098
- portal_parse_flags
 - portal.h, 2233
 - web/portal/flags.c, 2122
- portal_parse_json_str_array
 - portal.h, 2233
 - web/portal/parse.c, 2098
- portal_parse_sections
 - portal.h, 2234
 - web/portal/parse.c, 2099
- portal_print_login
 - portal.c, 2203
 - portal.h, 2234
- portal_process
 - portal.c, 2203
 - portal.h, 2235
- portal_settings_changepass
 - endpoint.c, 2196
- portal_settings_identity
 - endpoint.c, 2197
 - portal.h, 2235
- portal_smtp_create_data
 - mail.c, 2199
 - portal.h, 2235
- portal_smtp_merge_headers
 - mail.c, 2200
 - portal.h, 2236
- portal_smtp_relay_message
 - mail.c, 2200
 - portal.h, 2236
- portal_statistics_mutex
 - web/statistics/datatier.c, 718
- PORTAL_STATISTICS_TIMEOUT
 - web/statistics/datatier.c, 717
- portal_stats
 - web/statistics/datatier.c, 718
 - web/statistics/statistics.c, 806
- portal_upload
 - endpoint.c, 2197
 - portal.h, 2237
- portal_validate_request
 - endpoint.c, 2197
- position
 - tok_state_t, 160
- pref
 - mx_record_t, 114
- premium
 - magma_t, 98
 - relay_t, 126
- prev
 - cached_object, 32
- prime.c
 - org_key, 1899
 - org_signet, 1899
 - prime_alloc, 1896
 - prime_cleanup, 1896
 - prime_curve_group, 1899
 - prime_free, 1896
 - prime_get, 1896
 - prime_key_decrypt, 1896
 - prime_key_encrypt, 1897
 - prime_key_generate, 1897
 - prime_message_decrypt, 1897
 - prime_message_encrypt, 1897
 - prime_request_generate, 1897
 - prime_request_sign, 1897
 - prime_set, 1898
 - prime_signet_fingerprint, 1898
 - prime_signet_generate, 1898
 - prime_signet_validate, 1898
 - prime_start, 1898
 - prime_stop, 1899
- prime.h
 - __attribute__, 1906
 - AES_BLOCK_LEN, 1902
 - AES_KEY_LEN, 1902
 - AES_TAG_LEN, 1902
 - AES_VECTOR_LEN, 1902
 - ARMORED, 1904
 - BINARY, 1904
 - DEBUG, 1904
 - ED25519_ERR, 1904
 - ED25519_PRIV, 1904
 - ED25519_PUB, 1904
 - ED25519_KEY_PRIV_LEN, 1903
 - ED25519_KEY_PUB_LEN, 1903
 - ed25519_key_t, 1910
 - ed25519_key_type_t, 1904
 - ED25519_SIGNATURE_LEN, 1903
 - NONE, 1904
 - ORG_ENCRYPTION_KEY, 1906
 - ORG_FULL_SIGNATURE, 1906
 - ORG_IDENTIFIABLE_SIGNATURE, 1906
 - ORG_IDENTIFIER, 1906
 - ORG_PRIMARY_KEY, 1906
 - ORG_SECONDARY_KEY, 1906
 - ORG_SELF_SIGNATURE, 1906
 - org_key, 1910
 - org_signet, 1910

PRIME_BINARY_OBJECT, 1905
PRIME_CHUNK_BODY, 1905
PRIME_CHUNK_COMMON, 1905
PRIME_CHUNK_DESTINATION, 1905
PRIME_CHUNK_EPHEMERAL, 1905
PRIME_CHUNK_FLAG_ALTERNATE_ENCRYPT, 1905
PRIME_CHUNK_FLAG_ALTERNATE_PADDING, 1905
PRIME_CHUNK_FLAG_COMPRESSED, 1905
PRIME_CHUNK_FLAG_INVALID, 1905
PRIME_CHUNK_FLAG_NONE, 1905
PRIME_CHUNK_FLAG_SPANNING, 1905
PRIME_CHUNK_HEADERS, 1905
PRIME_CHUNK_INVALID, 1905
PRIME_CHUNK_ORIGIN, 1905
PRIME_CHUNK_TRACING, 1905
PRIME_MESSAGE_ABUSE, 1905
PRIME_MESSAGE_BOUNCE, 1905
PRIME_MESSAGE_DRAFT, 1905
PRIME_MESSAGE_ENCRYPTED, 1905
PRIME_MESSAGE_FORWARD, 1905
PRIME_MESSAGE_NAKED, 1905
PRIME_MESSAGE_SENT, 1905
PRIME_ORG_KEY, 1904
PRIME_ORG_KEY_ENCRYPTED, 1904
PRIME_ORG_SIGNET, 1904
PRIME_PROTOCOL_TICKET, 1905
PRIME_SIGNATURE_DESTINATION, 1905
PRIME_SIGNATURE_ORGIN, 1905
PRIME_SIGNATURE_TREE, 1905
PRIME_SIGNATURE_USER, 1905
PRIME_USER_KEY, 1904
PRIME_USER_KEY_ENCRYPTED, 1904
PRIME_USER_SIGNET, 1904
PRIME_USER_SIGNING_REQUEST, 1904
prime_alloc, 1907
prime_artifact_type_t, 1904
prime_chunk_keks_t, 1910
prime_chunk_keys_t, 1910
prime_chunk_slots_t, 1910
prime_cleanup, 1907
prime_encoding_t, 1904
prime_encrypted_chunk_t, 1911
prime_ephemeral_chunk_t, 1911
prime_flags_t, 1904
prime_free, 1907
prime_get, 1907
prime_key_decrypt, 1907
prime_key_encrypt, 1908
prime_key_generate, 1908
prime_message_chunk_flags_t, 1904
prime_message_chunk_type_t, 1905
prime_message_decrypt, 1908
prime_message_encrypt, 1908
prime_message_t, 1911
prime_message_type_t, 1905
prime_org_artifact_fields_t, 1905
prime_org_key_t, 1911
prime_org_signet_t, 1911
prime_request_generate, 1908
prime_request_sign, 1909
prime_set, 1909
prime_signature_tree_t, 1911
prime_signet_fingerprint, 1909
prime_signet_generate, 1909
prime_signet_validate, 1909
prime_start, 1909
prime_stop, 1910
prime_t, 1911
prime_type_t, 1904
prime_user_artifact_fields_t, 1906
prime_user_key_t, 1911
prime_user_signet_t, 1911
SECP256K1_ERR, 1906
SECP256K1_PRIV, 1906
SECP256K1_PUB, 1906
SECP256K1_KEY_PRIV_LEN, 1903
SECP256K1_KEY_PUB_LEN, 1903
secp256k1_key_t, 1904
secp256k1_key_type_t, 1906
SECP256K1_SHARED_SECRET_LEN, 1903
SECURITY, 1904
USER_ALTERNATE_ENCRYPTION_KEY, 1906
USER_CRYPTO_SIGNATURE, 1906
USER_CUSTODY_SIGNATURE, 1906
USER_ENCRYPTION_KEY, 1906
USER_FULL_SIGNATURE, 1906
USER_IDENTIFIABLE_SIGNATURE, 1906
USER_IDENTIFIER, 1906
USER_SELF_SIGNATURE, 1906
USER_SIGNING_KEY, 1906
prime/cryptography/cryptography.h
aes_artifact_decrypt, 1382
aes_artifact_encrypt, 1382
aes_chunk_decrypt, 1382
aes_chunk_encrypt, 1382
aes_cipher_key, 1382
aes_tag_shard, 1383
aes_vector_shard, 1383
ed25519_alloc, 1383
ed25519_free, 1384
ed25519_generate, 1384
ed25519_private_get, 1384
ed25519_private_set, 1384
ed25519_public_get, 1384
ed25519_public_set, 1384
ed25519_sign, 1385
ed25519_type, 1385
ed25519_verify, 1385
secp256k1_alloc, 1385
secp256k1_compute_kek, 1385
secp256k1_free, 1386
secp256k1_generate, 1386
secp256k1_private_get, 1386
secp256k1_private_set, 1387

- secp256k1_public_get, 1387
- secp256k1_public_set, 1387
- secp256k1_type, 1388
- prime/cryptography/ed25519.c
 - ed25519_alloc, 1666
 - ed25519_free, 1666
 - ed25519_generate, 1666
 - ed25519_private_get, 1666
 - ed25519_private_set, 1667
 - ed25519_public_get, 1667
 - ed25519_public_set, 1667
 - ed25519_sign, 1667
 - ed25519_type, 1667
 - ed25519_verify, 1667
- PRIME_BINARY_OBJECT
 - prime.h, 1905
- PRIME_CHUNK_BODY
 - prime.h, 1905
- PRIME_CHUNK_COMMON
 - prime.h, 1905
- PRIME_CHUNK_DESTINATION
 - prime.h, 1905
- PRIME_CHUNK_EPHEMERAL
 - prime.h, 1905
- PRIME_CHUNK_FLAG_ALTERNATE_ENCRYPT
 - prime.h, 1905
- PRIME_CHUNK_FLAG_ALTERNATE_PADDING
 - prime.h, 1905
- PRIME_CHUNK_FLAG_COMPRESSED
 - prime.h, 1905
- PRIME_CHUNK_FLAG_INVALID
 - prime.h, 1905
- PRIME_CHUNK_FLAG_NONE
 - prime.h, 1905
- PRIME_CHUNK_FLAG_SPANNING
 - prime.h, 1905
- PRIME_CHUNK_HEADERS
 - prime.h, 1905
- PRIME_CHUNK_INVALID
 - prime.h, 1905
- PRIME_CHUNK_ORIGIN
 - prime.h, 1905
- PRIME_CHUNK_TRACING
 - prime.h, 1905
- PRIME_MESSAGE_ABUSE
 - prime.h, 1905
- PRIME_MESSAGE_BOUNCE
 - prime.h, 1905
- PRIME_MESSAGE_DRAFT
 - prime.h, 1905
- PRIME_MESSAGE_ENCRYPTED
 - prime.h, 1905
- PRIME_MESSAGE_FORWARD
 - prime.h, 1905
- PRIME_MESSAGE_NAKED
 - prime.h, 1905
- PRIME_MESSAGE_SENT
 - prime.h, 1905
- PRIME_ORG_KEY
 - prime.h, 1904
- PRIME_ORG_KEY_ENCRYPTED
 - prime.h, 1904
- PRIME_ORG_SIGNET
 - prime.h, 1904
- PRIME_PROTOCOL_TICKET
 - prime.h, 1905
- PRIME_SIGNATURE_DESTINATION
 - prime.h, 1905
- PRIME_SIGNATURE_ORGIN
 - prime.h, 1905
- PRIME_SIGNATURE_TREE
 - prime.h, 1905
- PRIME_SIGNATURE_USER
 - prime.h, 1905
- PRIME_USER_KEY
 - prime.h, 1904
- PRIME_USER_KEY_ENCRYPTED
 - prime.h, 1904
- PRIME_USER_SIGNET
 - prime.h, 1904
- PRIME_USER_SIGNING_REQUEST
 - prime.h, 1904
- PRIME_ACTOR_AUTHOR
 - slots.c, 1890
- PRIME_ACTOR_DESTINATION
 - slots.c, 1890
- PRIME_ACTOR_NONE
 - slots.c, 1890
- PRIME_ACTOR_ORIGIN
 - slots.c, 1890
- PRIME_ACTOR_RECIPIENT
 - slots.c, 1891
- prime_alloc
 - prime.c, 1896
 - prime.h, 1907
- prime_artifact_type_t
 - prime.h, 1904
- PRIME_CHUNK_HEAD_LEN
 - aes.c, 1855
- prime_chunk_keks_t
 - prime.h, 1910
- PRIME_CHUNK_KEY_LEN
 - aes.c, 1855
- prime_chunk_keys_t
 - prime.h, 1910
- PRIME_CHUNK_PREFIX_LEN
 - aes.c, 1855
- prime_chunk_slots_t
 - prime.h, 1910
- prime_cleanup
 - prime.c, 1896
 - prime.h, 1907
- prime_curve_group
 - prime.c, 1899

- secp256k1.c, 1862
- prime_encoding_t
 - prime.h, 1904
- prime_encrypted_chunk_header_t
 - aes.c, 1858
- prime_encrypted_chunk_t
 - prime.h, 1911
- prime_encrypted_object_header_t
 - aes.c, 1858
- prime_encrypted_payload_prefix_t
 - aes.c, 1858
- prime_ephemeral_chunk_t
 - prime.h, 1911
- prime_field_data_t
 - binary.h, 1925
- prime_field_get
 - binary.h, 1925
 - fields.c, 1932
- prime_field_size_length
 - binary.h, 1926
 - fields.c, 1932
- prime_field_size_max
 - binary.h, 1926
 - fields.c, 1932
- prime_field_t
 - binary.h, 1931
- prime_field_type_t
 - binary.h, 1925
- prime_field_write
 - binary.h, 1926
 - fields.c, 1932
- PRIME_FIXED_SIZE
 - transposition.h, 1937
- prime_flags_t
 - prime.h, 1904
- prime_free
 - prime.c, 1896
 - prime.h, 1907
- prime_get
 - prime.c, 1896
 - prime.h, 1907
- prime_header_encrypted_message_write
 - binary.h, 1926
 - providers/prime/transposition/binary/headers.c, 1106
- prime_header_encrypted_org_key_write
 - binary.h, 1926
 - providers/prime/transposition/binary/headers.c, 1106
- prime_header_encrypted_user_key_write
 - binary.h, 1926
 - providers/prime/transposition/binary/headers.c, 1106
- prime_header_length
 - binary.h, 1927
 - providers/prime/transposition/binary/headers.c, 1106
- prime_header_org_key_write
 - binary.h, 1927
 - providers/prime/transposition/binary/headers.c, 1107
- prime_header_org_signet_write
 - binary.h, 1927
 - providers/prime/transposition/binary/headers.c, 1107
- prime_header_read
 - binary.h, 1927
 - providers/prime/transposition/binary/headers.c, 1107
- prime_header_user_key_write
 - binary.h, 1928
 - providers/prime/transposition/binary/headers.c, 1107
- prime_header_user_signet_write
 - binary.h, 1928
 - providers/prime/transposition/binary/headers.c, 1107
- prime_header_user_signing_request_write
 - binary.h, 1928
 - providers/prime/transposition/binary/headers.c, 1108
- prime_header_write
 - binary.h, 1928
 - providers/prime/transposition/binary/headers.c, 1108
- prime_key_decrypt
 - prime.c, 1896
 - prime.h, 1907
- prime_key_encrypt
 - prime.c, 1897
 - prime.h, 1908
- prime_key_generate
 - prime.c, 1897
 - prime.h, 1908
- PRIME_MAX_1_BYTE
 - transposition.h, 1937
- PRIME_MAX_2_BYTE
 - transposition.h, 1937
- PRIME_MAX_3_BYTE
 - transposition.h, 1937
- PRIME_MAX_4_BYTE
 - transposition.h, 1937
- prime_message_chunk_flags_t
 - prime.h, 1904
- prime_message_chunk_type_t
 - prime.h, 1905
- prime_message_decrypt
 - prime.c, 1897
 - prime.h, 1908
- prime_message_encrypt
 - prime.c, 1897
 - prime.h, 1908
- prime_message_t
 - prime.h, 1911
- prime_message_type_t
 - prime.h, 1905
- prime_object_alloc
 - binary.h, 1928
 - providers/prime/transposition/binary/objects.c, 1159
- prime_object_free
 - binary.h, 1929
 - providers/prime/transposition/binary/objects.c, 1159
- PRIME_OBJECT_HEAD_LEN
 - aes.c, 1856
- PRIME_OBJECT_KEY_LEN

- aes.c, 1856
- PRIME_OBJECT_PAYLOAD_PREFIX_LEN
 - aes.c, 1856
- PRIME_OBJECT_PREFIX_LEN
 - aes.c, 1856
- prime_object_size_max
 - binary.h, 1929
 - providers/prime/transposition/binary/objects.c, 1159
- prime_object_size_min
 - binary.h, 1929
 - providers/prime/transposition/binary/objects.c, 1159
- prime_object_t
 - binary.h, 1931
- prime_object_type
 - binary.h, 1929
 - providers/prime/transposition/binary/objects.c, 1159
- prime_org_artifact_fields_t
 - prime.h, 1905
- prime_org_key_t
 - prime.h, 1911
- prime_org_signet_t
 - prime.h, 1911
- prime_pem_begin
 - armored.h, 1920
 - pem.c, 1921
- prime_pem_end
 - armored.h, 1920
 - pem.c, 1922
- prime_pem_endings
 - pem.c, 1922
- prime_pem_headings
 - pem.c, 1922
- PRIME_PEM_LINE_WRAP_CHARS
 - pem.c, 1921
- PRIME_PEM_LINE_WRAP_LENGTH
 - pem.c, 1921
- PRIME_PEM_LINE_WRAP_TYPE
 - pem.c, 1921
- prime_pem_unwrap
 - armored.h, 1920
 - pem.c, 1922
- prime_pem_wrap
 - armored.h, 1920
 - pem.c, 1922
- prime_reader_open
 - binary.h, 1929
 - reader.c, 1934
- prime_reader_payload
 - binary.h, 1929
 - reader.c, 1934
- prime_reader_size
 - binary.h, 1930
 - reader.c, 1934
- prime_reader_t
 - binary.h, 1931
- prime_reader_type
 - binary.h, 1930
- reader.c, 1935
- prime_request_generate
 - prime.c, 1897
 - prime.h, 1908
- prime_request_sign
 - prime.c, 1897
 - prime.h, 1909
- prime_set
 - prime.c, 1898
 - prime.h, 1909
- prime_signature_tree_t
 - prime.h, 1911
- prime_signet_fingerprint
 - prime.c, 1898
 - prime.h, 1909
- prime_signet_generate
 - prime.c, 1898
 - prime.h, 1909
- prime_signet_validate
 - prime.c, 1898
 - prime.h, 1909
- prime_size_t
 - binary.h, 1925
- prime_start
 - prime.c, 1898
 - prime.h, 1909
- prime_stop
 - prime.c, 1899
 - prime.h, 1910
- prime_t
 - prime.h, 1911
- prime_type_t
 - prime.h, 1904
- prime_types
 - providers/prime/transposition/binary/objects.c, 1160
- prime_unpack
 - binary.h, 1930
 - unpack.c, 1936
- prime_unpack_fields
 - binary.h, 1930
 - unpack.c, 1936
- prime_unpack_validate
 - binary.h, 1931
 - unpack.c, 1936
- prime_user_artifact_fields_t
 - prime.h, 1906
- prime_user_key_t
 - prime.h, 1911
- prime_user_signet_t
 - prime.h, 1911
- print.c
 - st_aprint, 523
 - st_aprint_opts, 523
 - st_quick, 524
 - st_sprint, 524
 - st_vaprint_opts, 525
 - st_vsprint, 525

- st_write_variadic, 525
- private
 - key_pair_t, 84
- private_key
 - ED25519_KEY, 61
- prng
 - fuzz-curve25519.c, 1680
 - fuzz-ed25519.c, 1682
- process
 - status.c, 809
- process_find_pid
 - core/host/process.c, 314
 - host.h, 300
- process_kill
 - core/host/process.c, 314
 - host.h, 301
- process_maint
 - engine/context/process.c, 316
- process_my_pid
 - core/host/process.c, 314
 - host.h, 301
- process_start
 - context.h, 763
 - engine/context/process.c, 316
- process_stop
 - context.h, 764
 - engine/context/process.c, 317
- protocol
 - server_t, 131
- protocol.c
 - protocol_enqueue, 781
 - protocol_init, 781
 - protocol_process, 781
 - protocol_secure, 782
 - protocol_type, 782
- protocol_enqueue
 - protocol.c, 781
- protocol_init
 - controller.h, 778
 - protocol.c, 781
- protocol_process
 - controller.h, 778
 - protocol.c, 781
- protocol_secure
 - protocol.c, 782
- protocol_type
 - network/network.h, 950
 - protocol.c, 782
- providers.h
 - __attribute__, 1938
 - lib_load, 1938
 - lib_symbols, 1939
 - lib_unload, 1939
 - symbol_t, 1939
- providers/consumers/cache.c
 - cache_add, 619
 - cache_append, 619
 - cache_decrement, 619
 - cache_delete, 620
 - cache_flush, 620
 - cache_get, 620
 - cache_get_u64, 620
 - cache_increment, 621
 - cache_pool, 623
 - cache_set, 621
 - cache_set_u64, 622
 - cache_silent_add, 622
 - cache_start, 622
 - cache_stop, 622
 - lib_load_cache, 623
 - lib_version_cache, 623
- providers/consumers/counters.c
 - stamp_counter_check, 1100
 - stamp_counter_increment, 1100
- providers/dime/common/crypto.c
 - _compute_aes256_kek, 1199
 - _crypto_init, 1199
 - _crypto_shutdown, 1199
 - _decrypt_aes_256, 1199
 - _deserialize_ec_privkey, 1199
 - _deserialize_ec_pubkey, 1200
 - _deserialize_ed25519_privkey, 1200
 - _deserialize_ed25519_pubkey, 1200
 - _ec_sign_data, 1201
 - _ec_sign_sha_data, 1201
 - _ecies_env_derivation, 1201
 - _ed25519_sign_data, 1202
 - _ed25519_verify_sig, 1202
 - _encrypt_aes_256, 1202
 - _free_ec_key, 1202
 - _free_ed25519_key, 1203
 - _free_ed25519_key_chain, 1203
 - _generate_ec_keypair, 1203
 - _generate_ed25519_keypair, 1203
 - _get_random_bytes, 1203
 - _load_ec_privkey, 1204
 - _load_ec_pubkey, 1204
 - _load_ed25519_privkey, 1204
 - _serialize_ec_privkey, 1205
 - _serialize_ec_pubkey, 1205
 - _verify_ec_sha_signature, 1205
 - _verify_ec_signature, 1206
- providers/dime/common/network.h
 - _connect_timeout, 951
 - PUBLIC_FUNC_DECL, 951
- providers/dime/dmtp/commands.c
 - dime_dmtp_command_create, 1607
 - dime_dmtp_command_data, 1608
 - dime_dmtp_command_destroy, 1608
 - dime_dmtp_command_ehlo, 1608
 - dime_dmtp_command_format, 1608
 - dime_dmtp_command_helo, 1608
 - dime_dmtp_command_help, 1609
 - dime_dmtp_command_hist, 1609

- dime_dmtplib_command_mail, 1609
- dime_dmtplib_command_mode, 1610
- dime_dmtplib_command_noop, 1610
- dime_dmtplib_command_parse, 1610
- dime_dmtplib_command_quit, 1610
- dime_dmtplib_command_rcpt, 1611
- dime_dmtplib_command_rset, 1611
- dime_dmtplib_command_sgnt_domain, 1611
- dime_dmtplib_command_sgnt_user, 1611
- dime_dmtplib_command_starttls, 1612
- dime_dmtplib_command_vrfy_domain, 1612
- dime_dmtplib_command_vrfy_user, 1612
- dmtplib_command_list, 1613
- DMTP_EMPTY_ARG, 1607
- DMTP_IGNORE_ARG, 1607
- providers/dime/dmtplib/commands.h
 - dime_dmtplib_command_create, 1625
 - dime_dmtplib_command_data, 1625
 - dime_dmtplib_command_destroy, 1625
 - dime_dmtplib_command_ehlo, 1626
 - dime_dmtplib_command_format, 1626
 - dime_dmtplib_command_helo, 1626
 - dime_dmtplib_command_help, 1626
 - dime_dmtplib_command_hist, 1627
 - dime_dmtplib_command_mail, 1627
 - dime_dmtplib_command_mode, 1627
 - dime_dmtplib_command_noop, 1627
 - dime_dmtplib_command_parse, 1628
 - dime_dmtplib_command_quit, 1628
 - dime_dmtplib_command_rcpt, 1628
 - dime_dmtplib_command_rset, 1628
 - dime_dmtplib_command_sgnt_domain, 1629
 - dime_dmtplib_command_sgnt_user, 1629
 - dime_dmtplib_command_starttls, 1629
 - dime_dmtplib_command_vrfy_domain, 1630
 - dime_dmtplib_command_vrfy_user, 1630
 - DMTP_ARG_ANGULAR_BRCKT, 1624
 - DMTP_ARG_NONE, 1624
 - DMTP_ARG_PLAIN, 1624
 - DMTP_ARG_SQUARE_BRCKT, 1624
 - DMTP_COMMAND_INVALID, 1624
 - DMTP_DATA, 1624
 - DMTP_EHLO, 1624
 - DMTP_HELO, 1624
 - DMTP_HELP, 1624
 - DMTP_HIST, 1624
 - DMTP_MAIL, 1624
 - DMTP_MODE, 1624
 - DMTP_MODE_DMTP, 1625
 - dmtplib_mode_dmtp, 1625
 - dmtplib_mode_dual, 1625
 - dmtplib_mode_esmtplib, 1625
 - DMTP_MODE_NONE, 1625
 - DMTP_MODE_SMTP, 1625
 - dmtplib_mode_smtp, 1625
 - dmtplib_mode_unknown, 1625
 - DMTP_NOOP, 1624
 - DMTP_PARSE_ARGUMENT_ERROR, 1625
 - DMTP_PARSE_COMMAND_ERROR, 1625
 - DMTP_PARSE_INTERNAL_ERROR, 1625
 - DMTP_PARSE_INVALID_CALL, 1625
 - DMTP_PARSE_SUCCESS, 1625
 - DMTP_QUIT, 1624
 - DMTP_RCPT, 1624
 - DMTP_RSET, 1624
 - DMTP_SGNT, 1624
 - DMTP_STARTTLS, 1624
 - DMTP_VRFY, 1624
 - dmtplib_argument_type_t, 1624
 - dmtplib_command_list, 1630
 - dmtplib_command_type_t, 1624
 - DMTP_COMMANDS_NUM, 1624
 - DMTP_MAX_ARGUMENT_NUM, 1624
 - dmtplib_mode_t, 1624
 - dmtplib_parse_state_t, 1625
- providers/dime/signet-resolver/cache.c
 - _add_cached_object, 627
 - _add_cached_object_cmp, 627
 - _add_cached_object_cmp_forced, 628
 - _add_cached_object_forced, 628
 - _cached_object_exists, 629
 - _cached_object_exists_cmp, 629
 - _clone_cached_object, 629
 - _create_cached_object, 630
 - _deserialize_array, 630
 - _deserialize_array_cb, 631
 - _deserialize_data, 631
 - _deserialize_signet_cb, 632
 - _deserialize_str_array, 632
 - _deserialize_string, 632
 - _deserialize_vardata, 633
 - _destroy_cache_entry, 633
 - _destroy_signet_cb, 634
 - _dump_cache, 634
 - _dump_cache_data, 634
 - _dump_signet_cb, 634
 - _evict_if_stale, 635
 - _find_cached_object, 635
 - _find_cached_object_cmp, 635
 - _get_cache_location, 636
 - _get_cache_obj_data, 636
 - _get_cached_store_by_type, 636
 - _get_dime_dir_location, 637
 - _is_object_expired, 637
 - _load_cache_contents, 637
 - _lock_cache_store, 638
 - _mem_append_serialized, 638
 - _mem_append_serialized_array, 638
 - _mem_append_serialized_array_cb, 639
 - _mem_append_serialized_str_array, 639
 - _mem_append_serialized_string, 639
 - _remove_cached_object, 640
 - _remove_cached_object_cmp, 640
 - _replace_object, 641

- [_save_cache_contents](#), 641
- [_serialize_signet_cb](#), 641
- [_set_cache_location](#), 642
- [_set_cache_permissions](#), 642
- [_unlink_object](#), 642
- [_unlock_cache_store](#), 643
- [cached_stores](#), 643
- providers/dime/signet-resolver/cache.h
 - [_clone_cached_object](#), 652
 - [_create_cached_object](#), 652
 - [_deserialize_array](#), 653
 - [_deserialize_array_cb](#), 653
 - [_deserialize_data](#), 654
 - [_deserialize_signet_cb](#), 654
 - [_deserialize_str_array](#), 654
 - [_deserialize_string](#), 655
 - [_deserialize_vardata](#), 655
 - [_destroy_signet_cb](#), 656
 - [_dump_cache](#), 656
 - [_dump_cache_data](#), 656
 - [_dump_signet_cb](#), 657
 - [_evict_if_stale](#), 657
 - [_get_cached_store_by_type](#), 657
 - [_is_object_expired](#), 657
 - [_lock_cache_store](#), 658
 - [_mem_append_serialized](#), 658
 - [_mem_append_serialized_array](#), 658
 - [_mem_append_serialized_array_cb](#), 659
 - [_mem_append_serialized_str_array](#), 659
 - [_mem_append_serialized_string](#), 660
 - [_replace_object](#), 660
 - [_serialize_signet_cb](#), 660
 - [_unlink_object](#), 661
 - [_unlock_cache_store](#), 661
- [CACHE_PERM_ADD](#), 651
- [CACHE_PERM_ALL_FLAGS](#), 651
- [CACHE_PERM_DEFAULT](#), 651
- [CACHE_PERM_DELETE](#), 651
- [CACHE_PERM_LOAD](#), 651
- [CACHE_PERM_NONE](#), 651
- [CACHE_PERM_READ](#), 651
- [CACHE_PERM_SAVE](#), 651
- [cached_data_dnskey](#), 652
- [cached_data_drec](#), 652
- [cached_data_ds](#), 652
- [cached_data_ocsp](#), 652
- [cached_data_signet](#), 652
- [cached_data_unknown](#), 652
- [cached_data_type_t](#), 652
- [cached_object_t](#), 651
- [cached_store_comparator_t](#), 651
- [cached_stores](#), 662
- [custom_deserializer_t](#), 651
- [custom_dumper_t](#), 652
- [custom_serializer_t](#), 652
- [PUBLIC_FUNC_DECL](#), 661, 662
- providers/dime/signet-resolver/dmtp.c
 - [_dx_connect_dual](#), 1712
 - [_dx_connect_standard](#), 1713
 - [_get_signet](#), 1713
 - [_sgnt_resolv_destroy_dmtp_session](#), 1713
 - [_sgnt_resolv_dmtp_connect](#), 1714
 - [_sgnt_resolv_dmtp_data](#), 1714
 - [_sgnt_resolv_dmtp_ehlo](#), 1714
 - [_sgnt_resolv_dmtp_expect_banner](#), 1715
 - [_sgnt_resolv_dmtp_get_mode](#), 1715
 - [_sgnt_resolv_dmtp_get_signet](#), 1715
 - [_sgnt_resolv_dmtp_help](#), 1715
 - [_sgnt_resolv_dmtp_history](#), 1716
 - [_sgnt_resolv_dmtp_initiate_starttls](#), 1716
 - [_sgnt_resolv_dmtp_issue_command](#), 1716
 - [_sgnt_resolv_dmtp_mail_from](#), 1717
 - [_sgnt_resolv_dmtp_noop](#), 1717
 - [_sgnt_resolv_dmtp_quit](#), 1717
 - [_sgnt_resolv_dmtp_rcpt_to](#), 1718
 - [_sgnt_resolv_dmtp_reset](#), 1718
 - [_sgnt_resolv_dmtp_send_and_read](#), 1718
 - [_sgnt_resolv_dmtp_stats](#), 1718
 - [_sgnt_resolv_dmtp_str_to_mode](#), 1719
 - [_sgnt_resolv_dmtp_verify_signet](#), 1719
 - [_sgnt_resolv_dmtp_write_data](#), 1719
 - [_sgnt_resolv_parse_line_code](#), 1720
 - [_sgnt_resolv_read_dmtp_line](#), 1720
 - [_sgnt_resolv_read_dmtp_multiline](#), 1720
 - [_verify_dx_certificate](#), 1721
- providers/dime/signet-resolver/dmtp.h
 - [_sgnt_resolv_dmtp_expect_banner](#), 838
 - [_sgnt_resolv_dmtp_initiate_starttls](#), 839
 - [_sgnt_resolv_dmtp_issue_command](#), 839
 - [_sgnt_resolv_dmtp_send_and_read](#), 839
 - [_sgnt_resolv_dmtp_str_to_mode](#), 840
 - [_sgnt_resolv_dmtp_write_data](#), 840
 - [_sgnt_resolv_parse_line_code](#), 840
 - [_sgnt_resolv_read_dmtp_line](#), 841
 - [_sgnt_resolv_read_dmtp_multiline](#), 841
 - [data_type_7bit](#), 838
 - [data_type_8bit](#), 838
 - [data_type_default](#), 838
 - [DMTP_MODE_DMTP](#), 838
 - [dmtp_mode_dmtp](#), 838
 - [dmtp_mode_dual](#), 838
 - [dmtp_mode_esmtp](#), 838
 - [DMTP_MODE_NONE](#), 838
 - [DMTP_MODE_SMTP](#), 838
 - [dmtp_mode_smtp](#), 838
 - [dmtp_mode_unknown](#), 838
 - [DMTP_LINE_BUF_SIZE](#), 837
 - [dmtp_mail_datatype_t](#), 838
 - [dmtp_mail_rettype_t](#), 838
 - [DMTP_MAX_MX_RETRIES](#), 837
 - [dmtp_mode_t](#), 838
 - [DMTP_PORT](#), 837
 - [DMTP_PORT_DUAL](#), 837
 - [DMTP_V1_CIPHER_LIST](#), 837

- PUBLIC_FUNC_DECL, 842, 843
- return_type_default, 838
- return_type_display, 838
- return_type_full, 838
- return_type_header, 838
- providers/dime/signet/keys.c
 - dime_keys_enckey_fetch, 585
 - dime_keys_file_create, 585
 - dime_keys_generate, 586
 - dime_keys_signkey_fetch, 586
- providers/dime/signet/keys.h
 - dime_keys_enckey_fetch, 667
 - dime_keys_file_create, 667
 - dime_keys_generate, 668
 - dime_keys_signkey_fetch, 668
 - KEYS_TYPE_ERROR, 667
 - KEYS_TYPE_ORG, 667
 - KEYS_TYPE_USER, 667
 - keys_type_t, 667
- providers/parsers/parsers.h
 - lib_load_jansson, 458
 - lib_load_utf8proc, 458
 - lib_load_xml, 458
 - lib_version_jansson, 459
 - lib_version_utf8proc, 459
 - lib_version_xml, 459
 - utf8_error_string, 459
 - utf8_length_st, 460
 - utf8_valid_st, 460
 - xml_create_doc, 460
 - xml_create_parser_ctx, 461
 - xml_create_xpath_ctx, 461
 - xml_dump_doc, 461
 - xml_encode, 462
 - xml_error, 462
 - xml_free_doc, 462
 - xml_free_parser_ctx, 462
 - xml_free_xpath_ctx, 463
 - xml_free_xpath_obj, 463
 - xml_get_xpath_int16, 463
 - xml_get_xpath_int32, 464
 - xml_get_xpath_int64, 464
 - xml_get_xpath_int8, 464
 - xml_get_xpath_node_count, 464
 - xml_get_xpath_ns, 465
 - xml_get_xpath_st, 465
 - xml_get_xpath_uint16, 465
 - xml_get_xpath_uint32, 465
 - xml_get_xpath_uint64, 466
 - xml_get_xpath_uint8, 466
 - xml_node_add_sibling, 466
 - xml_node_free, 467
 - xml_node_get_content_st, 467
 - xml_node_new, 467
 - xml_node_set_content, 468
 - xml_node_set_property, 468
 - xml_set_xpath_ns, 468
 - xml_set_xpath_property, 468
 - xml_set_xpath_uint64, 469
 - xml_start, 469
 - xml_stop, 469
 - xml_xpath_eval, 470
 - xml_xpath_set_namespace, 470
- providers/prime/keys/keys.h
 - org_encrypted_key_get, 669
 - org_encrypted_key_set, 669
 - org_key_alloc, 669
 - org_key_free, 669
 - org_key_generate, 670
 - org_key_get, 670
 - org_key_length, 670
 - org_key_set, 670
 - user_encrypted_key_get, 670
 - user_encrypted_key_set, 670
 - user_key_alloc, 670
 - user_key_free, 671
 - user_key_generate, 671
 - user_key_get, 671
 - user_key_length, 671
 - user_key_set, 671
- providers/prime/messages/messages.c
 - encrypted_message_alloc, 1088
 - encrypted_message_cleanup, 1088
 - encrypted_message_free, 1088
 - naked_message_get, 1088
 - naked_message_set, 1088
- providers/prime/messages/messages.h
 - encrypted_message_alloc, 1186
 - encrypted_message_cleanup, 1186
 - encrypted_message_free, 1186
 - naked_message_get, 1186
 - naked_message_set, 1186
- providers/prime/transposition/binary/headers.c
 - prime_header_encrypted_message_write, 1106
 - prime_header_encrypted_org_key_write, 1106
 - prime_header_encrypted_user_key_write, 1106
 - prime_header_length, 1106
 - prime_header_org_key_write, 1107
 - prime_header_org_signet_write, 1107
 - prime_header_read, 1107
 - prime_header_user_key_write, 1107
 - prime_header_user_signet_write, 1107
 - prime_header_user_signing_request_write, 1108
 - prime_header_write, 1108
- providers/prime/transposition/binary/objects.c
 - prime_object_alloc, 1159
 - prime_object_free, 1159
 - prime_object_size_max, 1159
 - prime_object_size_min, 1159
 - prime_object_type, 1159
 - prime_types, 1160
- providers/stacie/crypto.c
 - stacie_decrypt, 1207
 - stacie_encrypt, 1207

- providers/storage/data.c
 - tank_delete_object, [499](#)
 - tank_insert_object, [499](#)
- ptr_chain_add
 - misc_pub.c, [1559](#)
- ptr_chain_clone
 - misc_pub.c, [1559](#)
- ptr_chain_free
 - misc_pub.c, [1559](#)
- pubkey
 - dime_record_t, [39](#)
 - dnskey, [57](#)
- public
 - key_pair_t, [84](#)
- PUBLIC_FUNC_DECL
 - dcrypto.h, [1518](#)
 - dns.h, [1758](#), [1759](#)
 - error.h, [1528](#)
 - misc.h, [1554](#)
 - mrec.h, [1767](#)
 - providers/dime/common/network.h, [951](#)
 - providers/dime/signet-resolver/cache.h, [661](#), [662](#)
 - providers/dime/signet-resolver/dmtp.h, [842](#), [843](#)
 - signet-ssl.h, [1774](#)
- PUBLIC_FUNC_DECL_VA
 - error.h, [1528](#)
 - misc.h, [1554](#)
- PUBLIC_FUNC_IMPL
 - error.h, [1528](#)
- PUBLIC_FUNC_IMPL_VA1
 - error.h, [1529](#)
- PUBLIC_FUNC_IMPL_VA1_RET
 - error.h, [1529](#)
- PUBLIC_FUNC_IMPL_VA2
 - error.h, [1529](#)
- PUBLIC_FUNC_IMPL_VA2_RET
 - error.h, [1529](#)
- PUBLIC_FUNC_IMPL_VOID
 - error.h, [1529](#)
- PUBLIC_FUNC_PROLOGUE
 - error.h, [1529](#)
- PUBLIC_FUNCTION_IMPLEMENT
 - error.h, [1529](#)
- PUBLIC_FUNCTION_IMPLEMENT_VOID
 - error.h, [1530](#)
- public_key
 - ED25519_KEY, [61](#)
- PUSH_ERROR
 - error.h, [1530](#)
- PUSH_ERROR_FMT
 - error.h, [1530](#)
- PUSH_ERROR_OPENSSL
 - error.h, [1530](#)
- PUSH_ERROR_RESOLVER
 - error.h, [1530](#)
- PUSH_ERROR_SYSCALL
 - error.h, [1531](#)
- qp.c
 - qp_decode, [254](#)
 - qp_encode, [254](#)
- qp_decode
 - encodings.h, [246](#)
 - qp.c, [254](#)
- qp_encode
 - encodings.h, [246](#)
 - qp.c, [254](#)
- QP_LINE_WRAP_LENGTH
 - encodings.h, [239](#)
- quarter
 - fuzz-curve25519.c, [1680](#)
 - fuzz-ed25519.c, [1681](#)
- queries
 - queries.h, [2082](#)
 - stmts.c, [1491](#)
- queries.h
 - AUTH_GET_BY_ADDRESS, [2071](#)
 - AUTH_GET_BY_USERID, [2071](#)
 - AUTH_UPDATE_LEGACY_TO_STACIE, [2071](#)
 - AUTH_UPDATE_USER_LOCK, [2071](#)
 - common, [2082](#)
 - DELETE_CONTACT, [2071](#)
 - DELETE_CONTACT_DETAILS, [2071](#)
 - DELETE_FOLDER, [2071](#)
 - DELETE_MESSAGE, [2071](#)
 - DELETE_MESSAGE_TAG, [2072](#)
 - DELETE_MESSAGE_TAGS, [2072](#)
 - DELETE_OBJECT, [2072](#)
 - DELETE_SIGNATURE, [2072](#)
 - DELETE_USER, [2072](#)
 - DELETE_USER_CONFIG, [2072](#)
 - FETCH_SIGNATURE, [2072](#)
 - INSERT_CONTACT, [2072](#)
 - INSERT_FOLDER, [2072](#)
 - INSERT_MESSAGE, [2072](#)
 - INSERT_MESSAGE_DUPLICATE, [2072](#)
 - INSERT_MESSAGE_TAG, [2073](#)
 - INSERT_OBJECT, [2073](#)
 - INSERT_RECEIVING, [2073](#)
 - INSERT_SIGNATURE, [2073](#)
 - INSERT_TRANSMITTING, [2073](#)
 - META_FETCH_MAIL_KEYS, [2073](#)
 - META_FETCH_SHARD, [2073](#)
 - META_FETCH_USER, [2073](#)
 - META_INSERT_MAIL_KEYS, [2073](#)
 - META_INSERT_SHARD, [2073](#)
 - queries, [2082](#)
 - QUERIES_INIT, [2073](#)
 - REGISTER_CHECK_USERNAME, [2074](#)
 - REGISTER_FETCH_BLOCKLIST, [2074](#)
 - REGISTER_INSERT_DISPATCH, [2074](#)
 - REGISTER_INSERT_FOLDER_NAME, [2074](#)
 - REGISTER_INSERT_LOG, [2074](#)
 - REGISTER_INSERT_MAILBOXES, [2074](#)
 - REGISTER_INSERT_PROFILE, [2074](#)

REGISTER_INSERT_STACIE_REALMS, 2074
 REGISTER_INSERT_STACIE_USER, 2074
 RENAME_FOLDER, 2075
 SELECT_AGENTS, 2075
 SELECT_ALERTS, 2075
 SELECT_ALL_MESSAGE_TAGS, 2075
 SELECT_AUTOREPLY, 2075
 SELECT_CONFIG, 2075
 SELECT_CONTACT_DETAILS, 2075
 SELECT_CONTACTS, 2075
 SELECT_DOMAINS, 2075
 SELECT_FILTERS, 2075
 SELECT_FOLDERS, 2076
 SELECT_HOST_NUMBER, 2076
 SELECT_MAILBOX_ADDRESS, 2076
 SELECT_MAILBOX_ADDRESS_ANY, 2076
 SELECT_MAILBOX_ALIASES, 2076
 SELECT_MESSAGE_FOLDER, 2076
 SELECT_MESSAGE_TAGS, 2076
 SELECT_MESSAGES, 2076
 SELECT_MESSAGES_ROLLOUT, 2076
 SELECT_PATTERNS, 2077
 SELECT_PREFS_INBOUND, 2077
 SELECT_RECEIVING, 2077
 SELECT_TRANSMITTING, 2077
 SELECT_USER, 2077
 SELECT_USER_AUTH, 2077
 SELECT_USER_CONFIG, 2077
 SELECT_USER_RECORD, 2077
 SELECT_USER_SALT, 2078
 SELECT_USER_STORAGE_KEYS, 2078
 SELECT_USERNUM_AUTH, 2078
 SELECT_USERNUM_AUTH_LEGACY, 2078
 SELECT_USERS_AUTH, 2078
 SMTP_SELECT_USER_AUTH, 2078
 STATISTICS_GET_EMAILS_RECEIVED_TODAY, 2078
 STATISTICS_GET_EMAILS_RECEIVED_WEEK, 2078
 STATISTICS_GET_EMAILS_SENT_TODAY, 2078
 STATISTICS_GET_EMAILS_SENT_WEEK, 2078
 STATISTICS_GET_TOTAL_USERS, 2079
 STATISTICS_GET_USERS_CHECKED_EMAIL_TODAY, 2079
 STATISTICS_GET_USERS_CHECKED_EMAIL_WEEK, 2079
 STATISTICS_GET_USERS_REGISTERED_TODAY, 2079
 STATISTICS_GET_USERS_REGISTERED_WEEK, 2079
 STATISTICS_GET_USERS_SENT_EMAIL_TODAY, 2079
 STATISTICS_GET_USERS_SENT_EMAIL_WEEK, 2079
 STMTS_INIT, 2079, 2082
 UPDATE_ALERTS_ACKNOWLEDGE, 2079
 UPDATE_CONTACT, 2079
 UPDATE_CONTACT_STAMP, 2080
 UPDATE_FOLDER, 2080
 UPDATE_LOG_IMAP, 2080
 UPDATE_LOG_POP, 2080
 UPDATE_LOG_RECEIVED, 2080
 UPDATE_LOG_SENT, 2080
 UPDATE_LOG_WEB, 2080
 UPDATE_MESSAGE_FLAGS_ADD, 2080
 UPDATE_MESSAGE_FLAGS_REMOVE, 2080
 UPDATE_MESSAGE_FLAGS_REPLACE, 2080
 UPDATE_MESSAGE_FOLDER, 2081
 UPDATE_MESSAGE_VISIBILITY, 2081
 UPDATE_SIGNATURE_FLAGS_ADD, 2081
 UPDATE_SIGNATURE_FLAGS_REMOVE, 2081
 UPDATE_USER_QUOTA_ADD, 2081
 UPDATE_USER_QUOTA_SUBTRACT, 2081
 UPDATE_USER_STORAGE_KEYS, 2081
 UPSERT_CONTACT_DETAIL, 2081
 UPSERT_USER_CONFIG, 2081
 QUERIES_INIT
 queries.h, 2073
 query.c
 sql_insert, 1472
 sql_insert_conn, 1472
 sql_num_rows, 1472
 sql_num_rows_conn, 1472
 sql_query, 1472
 sql_query_conn, 1473
 sql_query_res, 1473
 sql_query_res_conn, 1473
 sql_write, 1473
 sql_write_conn, 1474
 queue
 queue.c, 785
 queue.c
 dequeue, 783
 enqueue, 784
 items, 785
 lock, 785
 queue, 785
 queue_init, 784
 queue_shutdown, 784
 queue_signal, 785
 requeue, 785
 sema, 786
 workers, 786
 queue_init
 controller.h, 778
 queue.c, 784
 queue_shutdown
 controller.h, 779
 queue.c, 784
 queue_signal
 controller.h, 779
 queue.c, 785
 queue_t, 121
 data, 121
 function, 121
 next, 121
 requeue, 121
 quota
 smtp_inbound_prefs_t, 144

- r
 - batch_heap_t, 28
- RAND_bytes_d
 - symbols.c, 2004
 - symbols.h, 2053
- rand_choices
 - cryptography/cryptography.h, 1368
 - random.c, 1417
- RAND_cleanup_d
 - symbols.c, 2004
 - symbols.h, 2053
- rand_ctx
 - random.c, 1421
- rand_get_int16
 - cryptography/cryptography.h, 1369
 - random.c, 1418
- rand_get_int32
 - cryptography/cryptography.h, 1369
 - random.c, 1418
- rand_get_int64
 - cryptography/cryptography.h, 1369
 - random.c, 1418
- rand_get_int8
 - cryptography/cryptography.h, 1369
 - random.c, 1418
- rand_get_uint16
 - cryptography/cryptography.h, 1369
 - random.c, 1418
- rand_get_uint32
 - cryptography/cryptography.h, 1370
 - random.c, 1419
- rand_get_uint64
 - cryptography/cryptography.h, 1370
 - random.c, 1419
- rand_get_uint8
 - cryptography/cryptography.h, 1370
 - random.c, 1419
- RAND_load_file_d
 - symbols.c, 2004
 - symbols.h, 2053
- rand_start
 - cryptography/cryptography.h, 1371
 - random.c, 1420
- RAND_status_d
 - symbols.c, 2004
 - symbols.h, 2053
- rand_stop
 - cryptography/cryptography.h, 1371
 - random.c, 1420
- rand_thread_start
 - cryptography/cryptography.h, 1371
 - random.c, 1420
- rand_write
 - cryptography/cryptography.h, 1371
 - random.c, 1420
- random.c
 - rand_choices, 1417
 - rand_ctx, 1421
 - rand_get_int16, 1418
 - rand_get_int32, 1418
 - rand_get_int64, 1418
 - rand_get_int8, 1418
 - rand_get_uint16, 1418
 - rand_get_uint32, 1419
 - rand_get_uint64, 1419
 - rand_get_uint8, 1419
 - rand_start, 1420
 - rand_stop, 1420
 - rand_thread_start, 1420
 - rand_write, 1420
- random_data
 - fuzz-ed25519.c, 1682
- random_data_t, 122
 - m, 122
 - sk, 122
- range.c
 - imap_range_build, 2132
- rbl
 - smtp_inbound_prefs_t, 144
 - smtp_session_t, 153
- rblaction
 - smtp_inbound_prefs_t, 144
- rcptto
 - smtp_inbound_prefs_t, 144
- rdata
 - dnskey, 57
- rdlen
 - dnskey, 57
- read.c
 - client_read, 967
 - client_read_line, 967
 - con_read, 967
 - con_read_line, 968
- read_file_data
 - misc_pub.c, 1559
- read_pem_data
 - misc_pub.c, 1559
- reader.c
 - prime_reader_open, 1934
 - prime_reader_payload, 1934
 - prime_reader_size, 1934
 - prime_reader_type, 1935
- realms.c
 - stacie_realm_cipher, 1945
 - stacie_realm_key, 1945
 - stacie_realm_tag, 1946
 - stacie_realm_vector, 1946
- recent
 - imap_folder_status_t, 82
- recipient
 - dmime_object_t, 49
- recipient_limit
 - magma_t, 98
- recipients

- smtp_outbound_prefs_t, 149
- record_t
 - storage.h, 1964
- recp_keyslot
 - dmime_chunk_key_t, 42
- recv_size_limit
 - smtp_inbound_prefs_t, 144
- references.c
 - meta_user_ref_add, 1212
 - meta_user_ref_dec, 1212
 - meta_user_ref_protocol_total, 1213
 - meta_user_ref_stamp, 1213
 - meta_user_ref_total, 1213
- register.c
 - register_print_captcha, 2240
 - register_print_message, 2240
 - register_print_step1, 2241
 - register_print_step2, 2241
 - register_print_step3, 2241
 - register_process, 2242
- register.h
 - register_abuse_check_blocklist, 2245
 - register_abuse_checks, 2245
 - register_abuse_increment_history, 2245
 - register_blocklist_free, 2246
 - register_blocklist_update, 2246
 - register_business_step1, 2246
 - register_business_step2, 2246
 - register_business_validate_password, 2247
 - register_business_validate_username, 2247
 - register_captcha_generate, 2248
 - register_captcha_random_font, 2248
 - register_captcha_write_noise, 2248
 - register_data_check_username, 2248
 - register_data_fetch_blocklist, 2249
 - register_data_insert_user, 2249
 - REGISTER_PASSWORD_MAX_LENGTH, 2244
 - REGISTER_PASSWORD_MIN_LENGTH, 2244
 - register_print_captcha, 2250
 - register_print_message, 2250
 - register_print_step1, 2250
 - register_print_step2, 2251
 - register_print_step3, 2251
 - register_process, 2251
 - register_session_cache, 2252
 - register_session_free, 2252
 - register_session_generate, 2252
 - register_session_get, 2253
 - REGISTER_USERNAME_MAX_LENGTH, 2244
 - REGISTER_USERNAME_MIN_LENGTH, 2245
- register/abuse.c
 - register_abuse_check_blocklist, 2159
 - register_abuse_checks, 2159
 - register_abuse_increment_history, 2160
 - register_blocklist, 2160
 - register_blocklist_free, 2160
 - register_blocklist_lock, 2160
- register_blocklist_update, 2160
- register/business.c
 - register_business_step1, 2164
 - register_business_step2, 2164
 - register_business_validate_password, 2165
 - register_business_validate_username, 2165
- register_abuse_check_blocklist
 - register.h, 2245
 - register/abuse.c, 2159
- register_abuse_checks
 - register.h, 2245
 - register/abuse.c, 2159
- register_abuse_increment_history
 - register.h, 2245
 - register/abuse.c, 2160
- register_blocklist
 - register/abuse.c, 2160
- register_blocklist_free
 - register.h, 2246
 - register/abuse.c, 2160
- register_blocklist_lock
 - register/abuse.c, 2160
- register_blocklist_update
 - register.h, 2246
 - register/abuse.c, 2160
- register_business_step1
 - register.h, 2246
 - register/business.c, 2164
- register_business_step2
 - register.h, 2246
 - register/business.c, 2164
- register_business_validate_password
 - register.h, 2247
 - register/business.c, 2165
- register_business_validate_username
 - register.h, 2247
 - register/business.c, 2165
- register_captcha_generate
 - captcha.c, 2238
 - register.h, 2248
- register_captcha_random_font
 - captcha.c, 2238
 - register.h, 2248
- register_captcha_write_noise
 - captcha.c, 2238
 - register.h, 2248
- REGISTER_CHECK_USERNAME
 - queries.h, 2074
- register_data_check_username
 - register.h, 2248
 - web/register/datatier.c, 715
- register_data_fetch_blocklist
 - register.h, 2249
 - web/register/datatier.c, 715
- register_data_insert_user
 - register.h, 2249
 - web/register/datatier.c, 715

- REGISTER_FETCH_BLOCKLIST
 - queries.h, [2074](#)
- REGISTER_INSERT_DISPATCH
 - queries.h, [2074](#)
- REGISTER_INSERT_FOLDER_NAME
 - queries.h, [2074](#)
- REGISTER_INSERT_LOG
 - queries.h, [2074](#)
- REGISTER_INSERT_MAILBOXES
 - queries.h, [2074](#)
- REGISTER_INSERT_PROFILE
 - queries.h, [2074](#)
- REGISTER_INSERT_STACIE_REALMS
 - queries.h, [2074](#)
- REGISTER_INSERT_STACIE_USER
 - queries.h, [2074](#)
- REGISTER_PASSWORD_MAX_LENGTH
 - register.h, [2244](#)
- REGISTER_PASSWORD_MIN_LENGTH
 - register.h, [2244](#)
- register_print_captcha
 - register.c, [2240](#)
 - register.h, [2250](#)
- register_print_message
 - register.c, [2240](#)
 - register.h, [2250](#)
- register_print_step1
 - register.c, [2241](#)
 - register.h, [2250](#)
- register_print_step2
 - register.c, [2241](#)
 - register.h, [2251](#)
- register_print_step3
 - register.c, [2241](#)
 - register.h, [2251](#)
- register_process
 - register.c, [2242](#)
 - register.h, [2251](#)
- register_session_cache
 - register.h, [2252](#)
 - web/register/sessions.c, [1242](#)
- register_session_free
 - register.h, [2252](#)
 - web/register/sessions.c, [1242](#)
- register_session_generate
 - register.h, [2252](#)
 - web/register/sessions.c, [1243](#)
- register_session_get
 - register.h, [2253](#)
 - web/register/sessions.c, [1243](#)
- register_session_t, [123](#)
 - hvf_input, [123](#)
 - hvf_value, [123](#)
 - name, [123](#)
 - password, [123](#)
 - plan, [123](#)
 - username, [123](#)
 - usernum, [124](#)
- REGISTER_USERNAME_MAX_LENGTH
 - register.h, [2244](#)
- REGISTER_USERNAME_MIN_LENGTH
 - register.h, [2245](#)
- registration
 - magma_t, [98](#)
- REGISTRY
 - checkers.h, [1262](#)
- relaxed
 - cached_object, [32](#)
- relay
 - magma_t, [98](#)
- relay.h
 - relay_alloc, [742](#)
 - relay_config, [742](#)
 - relay_free, [743](#)
 - relay_output_help, [743](#)
 - relay_output_settings, [743](#)
 - relay_set_value, [744](#)
 - relay_validate, [744](#)
- relay_alloc
 - engine/config/relay/relay.c, [735](#)
 - relay.h, [742](#)
- relay_config
 - engine/config/relay/relay.c, [735](#)
 - relay.h, [742](#)
- relay_counter
 - engine/config/relay/relay.c, [736](#)
- relay_free
 - engine/config/relay/relay.c, [736](#)
 - relay.h, [743](#)
- relay_keys
 - engine/config/relay/keys.h, [665](#)
- relay_keys_t, [125](#)
 - description, [125](#)
 - name, [125](#)
 - norm, [125](#)
 - offset, [125](#)
 - required, [125](#)
- relay_limit
 - magma_t, [98](#)
- relay_output_help
 - engine/config/relay/relay.c, [736](#)
 - relay.h, [743](#)
- relay_output_settings
 - engine/config/relay/relay.c, [737](#)
 - relay.h, [743](#)
- relay_set_value
 - engine/config/relay/relay.c, [737](#)
 - relay.h, [744](#)
- relay_t, [126](#)
 - name, [126](#)
 - port, [126](#)
 - premium, [126](#)
 - secure, [126](#)
- relay_validate

- engine/config/relay/relay.c, [737](#)
- relay.h, [744](#)
- remaining
 - tok_state_t, [160](#)
- remove_cached_object
 - cache_pub.c, [1710](#)
- remove_cached_object_cmp
 - cache_pub.c, [1710](#)
- remove_message.c
 - mail_remove_message, [1164](#)
- RENAME_FOLDER
 - queries.h, [2075](#)
- replace.c
 - st_replace, [527](#)
 - st_swap, [527](#)
- request
 - __attribute__, [19](#)
- requests.c
 - user_request_generate, [1913](#)
 - user_request_get, [1913](#)
 - user_request_length, [1913](#)
 - user_request_rotation, [1913](#)
 - user_request_set, [1914](#)
 - user_request_sign, [1914](#)
 - user_request_verify_chain_of_custody, [1914](#)
 - user_request_verify_self, [1914](#)
- requeue
 - controller.h, [779](#)
 - queue.c, [785](#)
 - queue_t, [121](#)
- required
 - cache_keys_t, [29](#)
 - dmime_chunk_key_t, [42](#)
 - dmime_header_key_t, [45](#)
 - magma_keys_t, [87](#)
 - relay_keys_t, [125](#)
 - server_keys_t, [129](#)
 - signet_field_key_t, [135](#)
- res_bind_create
 - database.h, [1446](#)
 - results.c, [1476](#)
- res_bind_free
 - database.h, [1446](#)
 - results.c, [1476](#)
- res_field_block
 - database.h, [1446](#)
 - results.c, [1476](#)
- res_field_bool
 - database.h, [1447](#)
 - results.c, [1477](#)
- res_field_count
 - database.h, [1447](#)
 - results.c, [1477](#)
- res_field_double
 - database.h, [1447](#)
 - results.c, [1477](#)
- res_field_float
 - database.h, [1448](#)
 - results.c, [1478](#)
- res_field_generic
 - database.h, [1448](#)
 - results.c, [1478](#)
- res_field_int16
 - database.h, [1448](#)
 - results.c, [1478](#)
- res_field_int32
 - database.h, [1448](#)
 - results.c, [1478](#)
- res_field_int64
 - database.h, [1449](#)
 - results.c, [1479](#)
- res_field_int8
 - database.h, [1449](#)
 - results.c, [1479](#)
- res_field_length
 - database.h, [1449](#)
 - results.c, [1479](#)
- res_field_string
 - database.h, [1450](#)
 - results.c, [1480](#)
- res_field_uint16
 - database.h, [1450](#)
 - results.c, [1480](#)
- res_field_uint32
 - database.h, [1450](#)
 - results.c, [1480](#)
- res_field_uint64
 - database.h, [1451](#)
 - results.c, [1481](#)
- res_field_uint8
 - database.h, [1451](#)
 - results.c, [1481](#)
- res_row_count
 - database.h, [1451](#)
 - results.c, [1481](#)
- res_row_get
 - database.h, [1452](#)
 - results.c, [1482](#)
- res_row_next
 - database.h, [1452](#)
 - results.c, [1482](#)
- res_row_set
 - database.h, [1453](#)
 - results.c, [1483](#)
- res_row_store
 - database.h, [1453](#)
 - results.c, [1483](#)
- res_stmt_store
 - database.h, [1453](#)
 - results.c, [1483](#)
- res_table_alloc
 - database.h, [1453](#)
 - results.c, [1483](#)
- res_table_free

- database.h, [1453](#)
- results.c, [1483](#)
- RESERVED
 - checkers.h, [1262](#)
- resource
 - http_content_t, [75](#)
- response
 - __attribute__, [19](#)
- response.c
 - http_response, [2112](#)
 - http_response_allow_cross, [2112](#)
 - http_response_connection, [2113](#)
 - http_response_cookie, [2113](#)
 - http_response_header, [2113](#)
 - http_response_options, [2114](#)
 - http_response_status, [2114](#)
- results.c
 - res_bind_create, [1476](#)
 - res_bind_free, [1476](#)
 - res_field_block, [1476](#)
 - res_field_bool, [1477](#)
 - res_field_count, [1477](#)
 - res_field_double, [1477](#)
 - res_field_float, [1478](#)
 - res_field_generic, [1478](#)
 - res_field_int16, [1478](#)
 - res_field_int32, [1478](#)
 - res_field_int64, [1479](#)
 - res_field_int8, [1479](#)
 - res_field_length, [1479](#)
 - res_field_string, [1480](#)
 - res_field_uint16, [1480](#)
 - res_field_uint32, [1480](#)
 - res_field_uint64, [1481](#)
 - res_field_uint8, [1481](#)
 - res_row_count, [1481](#)
 - res_row_get, [1482](#)
 - res_row_next, [1482](#)
 - res_row_set, [1483](#)
 - res_row_store, [1483](#)
 - res_stmt_store, [1483](#)
 - res_table_alloc, [1483](#)
 - res_table_free, [1483](#)
- RET_ERROR_CUST
 - error.h, [1531](#)
- RET_ERROR_CUST_FMT
 - error.h, [1531](#)
- RET_ERROR_INT
 - error.h, [1531](#)
- RET_ERROR_INT_FMT
 - error.h, [1531](#)
- RET_ERROR_PTR
 - error.h, [1532](#)
- RET_ERROR_PTR_FMT
 - error.h, [1532](#)
- RET_ERROR_UINT
 - error.h, [1532](#)
- RET_ERROR_UINT_FMT
 - error.h, [1532](#)
- retry
 - magma_t, [98](#)
- return_type_default
 - providers/dime/signet-resolver/dmtp.h, [838](#)
- return_type_display
 - providers/dime/signet-resolver/dmtp.h, [838](#)
- return_type_full
 - providers/dime/signet-resolver/dmtp.h, [838](#)
- return_type_header
 - providers/dime/signet-resolver/dmtp.h, [838](#)
- reverse.c
 - con_reverse_check, [969](#)
 - con_reverse_domain, [969](#)
 - con_reverse_enqueue, [970](#)
 - con_reverse_lookup, [970](#)
 - con_reverse_status, [970](#)
- REVERSE_COMPLETE
 - network/network.h, [934](#)
- REVERSE_EMPTY
 - network/network.h, [934](#)
- REVERSE_ERROR
 - network/network.h, [934](#)
- REVERSE_PENDING
 - network/network.h, [934](#)
- rfc822
 - imap_fetch_dataitems_t, [79](#)
- rfc822_header
 - imap_fetch_dataitems_t, [79](#)
- rfc822_size
 - imap_fetch_dataitems_t, [79](#)
- rfc822_text
 - imap_fetch_dataitems_t, [79](#)
- rollout
 - smtp_inbound_prefs_t, [144](#)
- root
 - magma_t, [98](#)
- root_directory
 - magma_t, [99](#)
- ROOT_KEY_FILE
 - dns.h, [1749](#)
- ROTL32
 - ed25519-donna-portable.h, [1656](#)
- rotl32
 - fuzz-curve25519.c, [1680](#)
 - fuzz-ed25519.c, [1681](#)
- ROTR32
 - ed25519-donna-portable.h, [1656](#)
- row_t
 - database.h, [1445](#)
- rrsig_rr_t
 - dns.h, [1759](#)
- RSA_free_d
 - symbols.c, [2004](#)
 - symbols.h, [2053](#)
- RSA_new_d

- symbols.c, [2004](#)
- symbols.h, [2053](#)
- RSAPublicKey_dup_d
 - symbols.c, [2004](#)
 - symbols.h, [2053](#)
- ruenum
 - smtp_inbound_filter_t, [140](#)
- rwlock.c
 - rwlock_attr_destroy, [590](#)
 - rwlock_attr_getkind, [590](#)
 - rwlock_attr_init, [591](#)
 - rwlock_attr_setkind, [591](#)
 - rwlock_destroy, [592](#)
 - rwlock_init, [592](#)
 - rwlock_lock_read, [592](#)
 - rwlock_lock_write, [593](#)
 - rwlock_unlock, [593](#)
- rwlock_attr_destroy
 - rwlock.c, [590](#)
 - thread.h, [602](#)
- rwlock_attr_getkind
 - rwlock.c, [590](#)
 - thread.h, [602](#)
- rwlock_attr_init
 - rwlock.c, [591](#)
 - thread.h, [603](#)
- rwlock_attr_setkind
 - rwlock.c, [591](#)
 - thread.h, [603](#)
- rwlock_destroy
 - rwlock.c, [592](#)
 - thread.h, [603](#)
- rwlock_init
 - rwlock.c, [592](#)
 - thread.h, [604](#)
- rwlock_lock_read
 - rwlock.c, [592](#)
 - thread.h, [604](#)
- rwlock_lock_write
 - rwlock.c, [593](#)
 - thread.h, [604](#)
- rwlock_unlock
 - rwlock.c, [593](#)
 - thread.h, [605](#)
- S1_SWINDOWSIZE
 - ed25519-donna-impl-base.h, [1653](#)
 - ed25519-donna-impl-sse2.h, [1654](#)
- S1_TABLE_SIZE
 - ed25519-donna-impl-base.h, [1653](#)
 - ed25519-donna-impl-sse2.h, [1654](#)
- S2_SWINDOWSIZE
 - ed25519-donna-impl-base.h, [1653](#)
 - ed25519-donna-impl-sse2.h, [1654](#)
- S2_TABLE_SIZE
 - ed25519-donna-impl-base.h, [1653](#)
 - ed25519-donna-impl-sse2.h, [1654](#)
- S32
 - crc.c, [195](#)
- S8
 - crc.c, [195](#)
- s_free
 - sdsalloc.h, [1706](#)
- s_malloc
 - sdsalloc.h, [1706](#)
- s_realloc
 - sdsalloc.h, [1706](#)
- safeguard
 - magma_t, [99](#)
- salt
 - auth_t, [26](#)
 - magma_t, [99](#)
- sanity.c
 - sanity_check, [770](#)
- sanity_check
 - context.h, [764](#)
 - sanity.c, [770](#)
- save_cache_contents
 - cache_pub.c, [1710](#)
- scalars
 - batch_heap_t, [28](#)
- schema
 - magma_t, [99](#)
- scramble_alloc
 - deprecated.h, [1504](#)
 - deprecated/scramble.c, [1427](#)
- scramble_body_data
 - deprecated.h, [1504](#)
 - deprecated/scramble.c, [1428](#)
- scramble_body_hash
 - deprecated.h, [1504](#)
 - deprecated/scramble.c, [1428](#)
- scramble_body_length
 - deprecated.h, [1505](#)
 - deprecated/scramble.c, [1428](#)
- scramble_cleanup
 - deprecated.h, [1505](#)
 - deprecated/scramble.c, [1428](#)
- scramble_decrypt
 - deprecated.h, [1505](#)
 - deprecated/scramble.c, [1429](#)
- scramble_encrypt
 - deprecated.h, [1506](#)
 - deprecated/scramble.c, [1429](#)
- scramble_free
 - deprecated.h, [1506](#)
 - deprecated/scramble.c, [1429](#)
- scramble_head_t
 - cryptography/cryptography.h, [1379](#)
- scramble_import
 - deprecated.h, [1506](#)
 - deprecated/scramble.c, [1430](#)
- scramble_orig_length
 - deprecated.h, [1507](#)

- deprecated/scramble.c, 1430
- scramble_t
 - cryptography/cryptography.h, 1349
- scramble_total_length
 - deprecated.h, 1507
 - deprecated/scramble.c, 1430
- scramble_vector_data
 - deprecated.h, 1507
 - deprecated/scramble.c, 1431
- scramble_vector_length
 - deprecated.h, 1507
 - deprecated/scramble.c, 1431
- sds
 - sds.h, 1701
- sds.c
 - hex_digit_to_int, 1694
 - is_hex_digit, 1694
 - SDS_LLSTR_SIZE, 1694
 - sdsAllocPtr, 1694
 - sdsAllocSize, 1694
 - sdscat, 1694
 - sdscatfmt, 1694
 - sdscatlen, 1694
 - sdscatprintf, 1695
 - sdscatrepr, 1695
 - sdscatsds, 1695
 - sdscatvprintf, 1695
 - sdsclr, 1695
 - sdscmp, 1695
 - sdscopy, 1695
 - sdscopylen, 1695
 - sdsdup, 1695
 - sdsempty, 1696
 - sdsfree, 1696
 - sdsfreesplitres, 1696
 - sdsfromlonglong, 1696
 - sdsgrowzero, 1696
 - sdsIncrLen, 1696
 - sdsjoin, 1696
 - sdsjoinsds, 1696
 - sdsll2str, 1697
 - sdsMakeRoomFor, 1697
 - sdsmaphchars, 1697
 - sdsnew, 1697
 - sdsnewlen, 1697
 - sdsrange, 1697
 - sdsRemoveFreeSpace, 1697
 - sdssplitargs, 1697
 - sdssplitlen, 1697
 - sdstolower, 1698
 - sdstoupper, 1698
 - sdstrim, 1698
 - sdsull2str, 1698
 - sdsupdatelen, 1698
- sds.h
 - __attribute__, 1701
 - sds, 1701
 - SDS_HDR, 1700
 - SDS_HDR_VAR, 1700
 - SDS_MAX_PREALLOC, 1700
 - SDS_TYPE_16, 1700
 - SDS_TYPE_32, 1700
 - SDS_TYPE_5, 1700
 - SDS_TYPE_5_LEN, 1700
 - SDS_TYPE_64, 1700
 - SDS_TYPE_8, 1701
 - SDS_TYPE_BITS, 1701
 - SDS_TYPE_MASK, 1701
 - sdsAllocPtr, 1701
 - sdsAllocSize, 1701
 - sdscat, 1701
 - sdscatfmt, 1701
 - sdscatlen, 1702
 - sdscatprintf, 1702
 - sdscatrepr, 1702
 - sdscatsds, 1702
 - sdscatvprintf, 1702
 - sdsclr, 1702
 - sdscmp, 1702
 - sdscopy, 1702
 - sdscopylen, 1702
 - sdsdup, 1703
 - sdsempty, 1703
 - sdsfree, 1703
 - sdsfreesplitres, 1703
 - sdsfromlonglong, 1703
 - sdsgrowzero, 1703
 - sdsIncrLen, 1703
 - sdsjoin, 1703
 - sdsjoinsds, 1704
 - sdsMakeRoomFor, 1704
 - sdsmaphchars, 1704
 - sdsnew, 1704
 - sdsnewlen, 1704
 - sdsrange, 1704
 - sdsRemoveFreeSpace, 1704
 - sdssplitargs, 1704
 - sdssplitlen, 1704
 - sdstolower, 1705
 - sdstoupper, 1705
 - sdstrim, 1705
 - sdsupdatelen, 1705
- SDS_HDR
 - sds.h, 1700
- SDS_HDR_VAR
 - sds.h, 1700
- SDS_LLSTR_SIZE
 - sds.c, 1694
- SDS_MAX_PREALLOC
 - sds.h, 1700
- SDS_TYPE_16
 - sds.h, 1700
- SDS_TYPE_32
 - sds.h, 1700

SDS_TYPE_5
 sds.h, 1700
 SDS_TYPE_5_LEN
 sds.h, 1700
 SDS_TYPE_64
 sds.h, 1700
 SDS_TYPE_8
 sds.h, 1701
 SDS_TYPE_BITS
 sds.h, 1701
 SDS_TYPE_MASK
 sds.h, 1701
 sdsalloc.h
 s_free, 1706
 s_malloc, 1706
 s_realloc, 1706
 sdsAllocPtr
 sds.c, 1694
 sds.h, 1701
 sdsAllocSize
 sds.c, 1694
 sds.h, 1701
 sdscat
 sds.c, 1694
 sds.h, 1701
 sdscatfmt
 sds.c, 1694
 sds.h, 1701
 sdscatlen
 sds.c, 1694
 sds.h, 1702
 sdscatprintf
 sds.c, 1695
 sds.h, 1702
 sdscatrepr
 sds.c, 1695
 sds.h, 1702
 sdscatsds
 sds.c, 1695
 sds.h, 1702
 sdscatvprintf
 sds.c, 1695
 sds.h, 1702
 sdsclear
 sds.c, 1695
 sds.h, 1702
 sdscmp
 sds.c, 1695
 sds.h, 1702
 sdscopy
 sds.c, 1695
 sds.h, 1702
 sdscopylen
 sds.c, 1695
 sds.h, 1702
 sdsdup
 sds.c, 1695
 sds.h, 1703
 sdsempty
 sds.c, 1696
 sds.h, 1703
 sdsfree
 sds.c, 1696
 sds.h, 1703
 sdsfreesplitres
 sds.c, 1696
 sds.h, 1703
 sdsfromlonglong
 sds.c, 1696
 sds.h, 1703
 sdsgrowzero
 sds.c, 1696
 sds.h, 1703
 sdsIncrLen
 sds.c, 1696
 sds.h, 1703
 sdsjoin
 sds.c, 1696
 sds.h, 1703
 sdsjoinsds
 sds.c, 1696
 sds.h, 1704
 sdsl2str
 sds.c, 1697
 sdsMakeRoomFor
 sds.c, 1697
 sds.h, 1704
 sdsmaphchars
 sds.c, 1697
 sds.h, 1704
 sdsnew
 sds.c, 1697
 sds.h, 1704
 sdsnewlen
 sds.c, 1697
 sds.h, 1704
 sdsrange
 sds.c, 1697
 sds.h, 1704
 sdsRemoveFreeSpace
 sds.c, 1697
 sds.h, 1704
 sdssplitargs
 sds.c, 1697
 sds.h, 1704
 sdssplitlen
 sds.c, 1697
 sds.h, 1704
 sdstolower
 sds.c, 1698
 sds.h, 1705
 sdstoupper
 sds.c, 1698
 sds.h, 1705

- sdstrim
 - sds.c, 1698
 - sds.h, 1705
- sdsull2str
 - sds.c, 1698
- sdsupdatelen
 - sds.c, 1698
 - sds.h, 1705
- seasoning
 - auth_t, 26
- secp256k1.c
 - prime_curve_group, 1862
 - secp256k1_alloc, 1859
 - secp256k1_compute_kek, 1859
 - secp256k1_free, 1860
 - secp256k1_generate, 1860
 - secp256k1_private_get, 1860
 - secp256k1_private_set, 1861
 - secp256k1_public_get, 1861
 - secp256k1_public_set, 1861
 - secp256k1_type, 1862
- SECP256K1_ERR
 - prime.h, 1906
- SECP256K1_PRIV
 - prime.h, 1906
- SECP256K1_PUB
 - prime.h, 1906
- secp256k1_alloc
 - prime/cryptography/cryptography.h, 1385
 - secp256k1.c, 1859
- secp256k1_compute_kek
 - prime/cryptography/cryptography.h, 1385
 - secp256k1.c, 1859
- secp256k1_free
 - prime/cryptography/cryptography.h, 1386
 - secp256k1.c, 1860
- secp256k1_generate
 - prime/cryptography/cryptography.h, 1386
 - secp256k1.c, 1860
- SECP256K1_KEY_PRIV_LEN
 - prime.h, 1903
- SECP256K1_KEY_PUB_LEN
 - prime.h, 1903
- secp256k1_key_t
 - prime.h, 1904
- secp256k1_key_type_t
 - prime.h, 1906
- secp256k1_private_get
 - prime/cryptography/cryptography.h, 1386
 - secp256k1.c, 1860
- secp256k1_private_set
 - prime/cryptography/cryptography.h, 1387
 - secp256k1.c, 1861
- secp256k1_public_get
 - prime/cryptography/cryptography.h, 1387
 - secp256k1.c, 1861
- secp256k1_public_set
 - prime/cryptography/cryptography.h, 1387
 - secp256k1.c, 1861
- SECP256K1_SHARED_SECRET_LEN
 - prime.h, 1903
- secp256k1_type
 - prime/cryptography/cryptography.h, 1388
 - secp256k1.c, 1862
- section
 - dmime_chunk_key_t, 42
- SECURE
 - strings.h, 543
- secure
 - magma_t, 99
 - relay_t, 126
 - smtp_inbound_prefs_t, 145
- secure.c
 - __attribute__, 387
 - allocated, 390
 - bytes, 390
 - data, 390
 - data_true, 390
 - enabled, 391
 - items, 391
 - length, 391
 - length_true, 391
 - lock, 391
 - MM_SEC_CHUNK_ALLOCATED, 387
 - MM_SEC_CHUNK_AVAILABLE, 387
 - mm_sec_alloc, 387
 - mm_sec_chunk_merge, 387
 - mm_sec_chunk_new, 387
 - mm_sec_chunk_next, 387
 - mm_sec_chunk_prev, 388
 - mm_sec_cleanup, 388
 - mm_sec_free, 388
 - mm_sec_realloc, 389
 - mm_sec_secured, 389
 - mm_sec_start, 389
 - mm_sec_stats, 389
 - mm_sec_stop, 390
 - secured_t, 391
 - slab, 391
- secure_wipe
 - misc_pub.c, 1559
- secured_t
 - secure.c, 391
- SECURITY
 - prime.h, 1904
- seed_length
 - magma_t, 99
- segment_t
 - host.h, 287
- SELECT_AGENTS
 - queries.h, 2075
- SELECT_ALERTS
 - queries.h, 2075
- SELECT_ALL_MESSAGE_TAGS

- queries.h, [2075](#)
- SELECT_AUTOREPLY
 - queries.h, [2075](#)
- SELECT_CONFIG
 - queries.h, [2075](#)
- SELECT_CONTACT_DETAILS
 - queries.h, [2075](#)
- SELECT_CONTACTS
 - queries.h, [2075](#)
- SELECT_DOMAINS
 - queries.h, [2075](#)
- SELECT_FILTERS
 - queries.h, [2075](#)
- SELECT_FOLDERS
 - queries.h, [2076](#)
- SELECT_HOST_NUMBER
 - queries.h, [2076](#)
- SELECT_MAILBOX_ADDRESS
 - queries.h, [2076](#)
- SELECT_MAILBOX_ADDRESS_ANY
 - queries.h, [2076](#)
- SELECT_MAILBOX_ALIASES
 - queries.h, [2076](#)
- SELECT_MESSAGE_FOLDER
 - queries.h, [2076](#)
- SELECT_MESSAGE_TAGS
 - queries.h, [2076](#)
- SELECT_MESSAGES
 - queries.h, [2076](#)
- SELECT_MESSAGES_ROLLOUT
 - queries.h, [2076](#)
- SELECT_PATTERNS
 - queries.h, [2077](#)
- SELECT_PREFS_INBOUND
 - queries.h, [2077](#)
- SELECT_RECEIVING
 - queries.h, [2077](#)
- SELECT_TRANSMITTING
 - queries.h, [2077](#)
- SELECT_USER
 - queries.h, [2077](#)
- SELECT_USER_AUTH
 - queries.h, [2077](#)
- SELECT_USER_CONFIG
 - queries.h, [2077](#)
- SELECT_USER_RECORD
 - queries.h, [2077](#)
- SELECT_USER_SALT
 - queries.h, [2078](#)
- SELECT_USER_STORAGE_KEYS
 - queries.h, [2078](#)
- SELECT_USERNUM_AUTH
 - queries.h, [2078](#)
- SELECT_USERNUM_AUTH_LEGACY
 - queries.h, [2078](#)
- SELECT_USERS_AUTH
 - queries.h, [2078](#)

- selector
 - magma_t, [99](#)
- sema
 - queue.c, [786](#)
- send_size_limit
 - smtp_outbound_prefs_t, [149](#)
- sender
 - magma_t, [99](#)
- sent_today
 - smtp_outbound_prefs_t, [149](#)
- sequential
 - dmime_chunk_key_t, [42](#)
- serial_get
 - objects.h, [1227](#)
 - serials.c, [1217](#)
- serial_increment
 - objects.h, [1227](#)
 - serials.c, [1217](#)
- serial_prefix
 - serials.c, [1218](#)
- serial_prefix_strings
 - serials.c, [1219](#)
- serial_reset
 - objects.h, [1227](#)
 - serials.c, [1218](#)
- serialization.c
 - serialize_int16, [1326](#)
 - serialize_int32, [1327](#)
 - serialize_int64, [1327](#)
 - serialize_ns, [1327](#)
 - serialize_ssz, [1328](#)
 - serialize_st, [1328](#)
 - serialize_sz, [1328](#)
 - serialize_uint16, [1329](#)
 - serialize_uint32, [1329](#)
 - serialize_uint64, [1330](#)
- serialization_t
 - consumers.h, [1320](#)
- serialize
 - cached_store_t, [35](#)
- serialize_ec_privkey
 - crypto_pub.c, [1513](#)
- serialize_ec_pubkey
 - crypto_pub.c, [1513](#)
- serialize_int16
 - consumers.h, [1316](#)
 - serialization.c, [1326](#)
- serialize_int32
 - consumers.h, [1316](#)
 - serialization.c, [1327](#)
- serialize_int64
 - consumers.h, [1317](#)
 - serialization.c, [1327](#)
- serialize_ns
 - consumers.h, [1317](#)
 - serialization.c, [1327](#)
- serialize_ssz

- consumers.h, [1318](#)
- serialization.c, [1328](#)
- serialize_st
 - consumers.h, [1318](#)
 - serialization.c, [1328](#)
- serialize_sz
 - consumers.h, [1318](#)
 - serialization.c, [1328](#)
- serialize_uint16
 - consumers.h, [1319](#)
 - serialization.c, [1329](#)
- serialize_uint32
 - consumers.h, [1319](#)
 - serialization.c, [1329](#)
- serialize_uint64
 - consumers.h, [1320](#)
 - serialization.c, [1330](#)
- serials.c
 - serial_get, [1217](#)
 - serial_increment, [1217](#)
 - serial_prefix, [1218](#)
 - serial_prefix_strings, [1219](#)
 - serial_reset, [1218](#)
- serv_charset
 - mysql.c, [1471](#)
- serv_charset_mysql
 - database.h, [1454](#)
 - mysql.c, [1467](#)
- serv_schema
 - mysql.c, [1471](#)
- serv_schema_mysql
 - database.h, [1454](#)
 - mysql.c, [1467](#)
- serv_type_mysql
 - database.h, [1454](#)
 - mysql.c, [1467](#)
- serv_version
 - mysql.c, [1471](#)
- serv_version_mysql
 - database.h, [1454](#)
 - mysql.c, [1468](#)
- server_config_t, [127](#)
 - bad_command_cutoff, [127](#)
 - bad_command_delay, [127](#)
 - banner, [127](#)
 - certificate, [127](#)
 - domain, [127](#)
 - listen_queue, [127](#)
 - name, [128](#)
 - next, [128](#)
 - normal_sd, [128](#)
 - socket_timeout, [128](#)
 - spam, [128](#)
 - tcp_port, [128](#)
 - tls_ctx, [128](#)
 - tls_port, [128](#)
 - tls_sd, [128](#)
- server_keys
 - engine/config/servers/keys.h, [666](#)
- server_keys_t, [129](#)
 - description, [129](#)
 - name, [129](#)
 - norm, [129](#)
 - offset, [129](#)
 - required, [129](#)
- server_t, [130](#)
 - certificate, [130](#)
 - context, [130](#)
 - cutoff, [130](#)
 - delay, [130](#)
 - domain, [131](#)
 - enabled, [131](#)
 - ipv6, [131](#)
 - listen_queue, [131](#)
 - name, [131](#)
 - network, [131](#)
 - port, [131](#)
 - protocol, [131](#)
 - sockd, [131](#)
 - timeout, [132](#)
 - tls, [132](#)
 - type, [132](#)
 - violations, [132](#)
- servers
 - magma_t, [99](#)
- servers.c
 - servers_alloc, [747](#)
 - servers_config, [747](#)
 - servers_encryption_start, [747](#)
 - servers_encryption_stop, [747](#)
 - servers_free, [748](#)
 - servers_get_by_protocol, [748](#)
 - servers_get_by_socket, [748](#)
 - servers_get_count_using_port, [749](#)
 - servers_network_start, [749](#)
 - servers_network_stop, [749](#)
 - servers_output_help, [750](#)
 - servers_output_settings, [750](#)
 - servers_set_value, [750](#)
 - servers_validate, [751](#)
- servers/dmtp/commands.c
 - dmtp_compare, [1614](#)
 - dmtp_process, [1614](#)
 - dmtp_requeue, [1614](#)
 - dmtp_sort, [1614](#)
- servers/dmtp/commands.h
 - dmtp_commands, [1631](#)
- servers/dmtp/dmtp.c
 - dmtp_data, [1722](#)
 - dmtp_ehlo, [1723](#)
 - dmtp_helo, [1723](#)
 - dmtp_help, [1723](#)
 - dmtp_hist, [1723](#)
 - dmtp_init, [1723](#)

- dmtp_invalid, 1724
- dmtp_mail, 1724
- dmtp_mode, 1724
- dmtp_noop, 1724
- dmtp_quit, 1725
- dmtp_rcpt, 1725
- dmtp_rset, 1725
- dmtp_sgnt, 1725
- dmtp_verb, 1726
- dmtp_vrfy, 1726
- servers/dmtp/dmtp.h
 - dmtp_compare, 845
 - dmtp_data, 845
 - dmtp_ehlo, 845
 - dmtp_helo, 845
 - dmtp_help, 845
 - dmtp_hist, 846
 - dmtp_init, 846
 - dmtp_invalid, 846
 - dmtp_mail, 846
 - dmtp_mode, 846
 - dmtp_noop, 847
 - dmtp_process, 847
 - dmtp_quit, 847
 - dmtp_rcpt, 847
 - dmtp_requeue, 848
 - dmtp_rset, 848
 - dmtp_session_destroy, 848
 - dmtp_session_reset, 848
 - dmtp_sgnt, 849
 - dmtp_sort, 849
 - dmtp_verb, 849
 - dmtp_vrfy, 849
- servers/http/data.c
 - http_data_free, 500
 - http_data_get, 500
 - http_data_header_parse, 501
 - http_data_header_parse_line, 501
 - http_data_value_decode, 501
 - http_data_value_parse, 502
- servers/http/errors.c
 - http_print_301, 270
 - http_print_400, 271
 - http_print_403, 271
 - http_print_404, 271
 - http_print_405, 271
 - http_print_500, 272
 - http_print_500_log, 272
 - http_print_501, 272
- servers/http/http.h
 - get_header_opt, 856
 - get_header_value_noopt, 856
 - HTTP_CLOSE, 855
 - HTTP_COMPLETE, 855
 - HTTP_CONNECTION_CLOSE, 855
 - HTTP_CONNECTION_KEEPALIVE, 855
 - HTTP_CONNECTION_NEUTRAL, 855
 - HTTP_COOKIE_DELETE, 855
 - HTTP_COOKIE_NEUTRAL, 855
 - HTTP_COOKIE_SET, 855
 - HTTP_ERROR_400, 855
 - HTTP_ERROR_401, 855
 - HTTP_ERROR_403, 855
 - HTTP_ERROR_404, 855
 - HTTP_ERROR_405, 855
 - HTTP_ERROR_422, 855
 - HTTP_ERROR_500, 855
 - HTTP_ERROR_501, 855
 - HTTP_OK, 855
 - HTTP_PARSE_COOKIE, 855
 - HTTP_PARSE_HEADER, 855
 - HTTP_PARSE_PAIRS, 855
 - HTTP_READ_BODY, 855
 - HTTP_READY, 855
 - HTTP_RESPOND, 855
 - http_body, 856
 - http_close, 857
 - http_content_load_directory, 857
 - http_content_load_fonts, 857
 - http_content_refresh, 857
 - http_content_start, 858
 - http_content_stop, 858
 - http_data_free, 858
 - http_data_get, 859
 - http_data_header_parse, 859
 - http_data_header_parse_line, 859
 - http_data_value_decode, 860
 - http_data_value_parse, 860
 - http_free_content, 860
 - http_get_static, 861
 - http_get_template, 861
 - http_init, 861
 - http_load_file, 862
 - http_page_free, 862
 - http_page_get, 862
 - http_parse_context, 863
 - http_parse_header, 863
 - http_parse_method, 864
 - http_parse_origin, 864
 - http_parse_pairs, 865
 - http_print_301, 865
 - http_print_400, 865
 - http_print_403, 866
 - http_print_404, 866
 - http_print_405, 866
 - http_print_500, 867
 - http_print_500_log, 867
 - http_print_501, 867
 - http_process, 867
 - http_requeue, 868
 - http_response, 868
 - http_response_allow_cross, 869
 - http_response_connection, 869
 - http_response_cookie, 869

- http_response_header, 869
- http_response_options, 870
- http_response_status, 870
- http_session_destroy, 870
- http_session_reset, 871
- multipart_get_boundary, 871
- servers/http/parse.c
 - get_header_opt, 2084
 - get_header_value_noopt, 2084
 - http_parse_context, 2085
 - http_parse_header, 2085
 - http_parse_method, 2086
 - http_parse_origin, 2086
 - http_parse_pairs, 2086
 - multipart_get_boundary, 2087
- servers/http/sessions.c
 - http_session_destroy, 1238
 - http_session_reset, 1238
- servers/imap/commands.c
 - imap_compare, 1616
 - imap_process, 1616
 - imap_requeue, 1617
 - imap_sort, 1617
- servers/imap/commands.h
 - imap_commands, 1633
- servers/imap/flags.c
 - imap_flag_action, 2121
 - imap_flag_parse, 2121
 - imap_get_flag, 2121
 - imap_update_flags, 2121
- servers/imap/folders.c
 - imap_count_folder_levels, 1065
 - imap_folder_compare, 1065
 - imap_folder_create, 1066
 - imap_folder_name_escaped, 1066
 - imap_folder_remove, 1066
 - imap_folder_rename, 1067
 - imap_folder_status, 1067
 - imap_narrow_folders, 1068
 - imap_next_folder_order, 1068
 - imap_valid_folder_name, 1068
- servers/imap/imap.h
 - imap_append, 879
 - imap_append_message, 879
 - IMAP_ARGUMENT_TYPE_ARRAY, 877
 - IMAP_ARGUMENT_TYPE_ASTRING, 877
 - IMAP_ARGUMENT_TYPE_EMPTY, 877
 - IMAP_ARGUMENT_TYPE_LITERAL, 877
 - IMAP_ARGUMENT_TYPE_NSTRING, 877
 - IMAP_ARGUMENT_TYPE_QSTRING, 877
 - IMAP_ARRAY_RECURSION_LIMIT, 877
 - imap_build_array, 879
 - imap_build_array_isliteral, 879
 - imap_capability, 880
 - imap_check, 880
 - imap_close, 880
 - imap_command_parser, 880
 - imap_compare, 880
 - imap_copy, 881
 - imap_count_folder_levels, 881
 - imap_create, 881
 - imap_delete, 882
 - imap_duplicate_messages, 882
 - imap_examine, 882
 - imap_expunge, 883
 - imap_fetch, 883
 - imap_fetch_body, 883
 - IMAP_FETCH_BODY_HEADER, 877
 - imap_fetch_body_header, 883
 - IMAP_FETCH_BODY_HEADER_FIELDS, 878
 - IMAP_FETCH_BODY_HEADER_FIELDS_NOT, 878
 - IMAP_FETCH_BODY_MIME, 878
 - imap_fetch_body_mime, 883
 - IMAP_FETCH_BODY_PART, 878
 - imap_fetch_body_part, 884
 - imap_fetch_body_portion, 884
 - imap_fetch_body_tag, 884
 - IMAP_FETCH_BODY_TEXT, 878
 - imap_fetch_bodystructure, 884
 - imap_fetch_envelope, 884
 - imap_fetch_free_items, 884
 - imap_fetch_message, 885
 - imap_fetch_parse_partial, 885
 - imap_fetch_response_add, 885
 - imap_fetch_response_free, 885
 - imap_fetch_return_header, 885
 - imap_fetch_return_message, 885
 - imap_fetch_return_mime, 886
 - imap_fetch_return_text, 886
 - imap_flag_action, 886
 - IMAP_FLAG_ADD, 878
 - imap_flag_parse, 886
 - IMAP_FLAG_REMOVE, 878
 - IMAP_FLAG_REPLACE, 878
 - IMAP_FLAG_SILENT, 878
 - imap_folder_compare, 886
 - imap_folder_create, 886
 - imap_folder_name_escaped, 887
 - IMAP_FOLDER_RECURSION_LIMIT, 878
 - imap_folder_remove, 887
 - imap_folder_rename, 888
 - imap_folder_status, 888
 - imap_get_ar_ar, 889
 - imap_get_flag, 889
 - imap_get_ptr, 889
 - imap_get_st_ar, 889
 - imap_get_type_ar, 890
 - imap_id, 890
 - imap_idle, 890
 - imap_init, 890
 - imap_invalid, 891
 - imap_list, 891
 - imap_login, 891
 - imap_logout, 892

- imap_lsub, 892
- imap_message_copier, 892
- imap_message_expunge, 892
- imap_narrow_folders, 892
- imap_narrow_messages, 893
- imap_next_folder_order, 893
- imap_noop, 893
- imap_parse_address, 893
- imap_parse_address_breaker, 893
- imap_parse_address_part, 894
- imap_parse_address_put, 894
- imap_parse_arguments, 894
- imap_parse_array, 894
- imap_parse_astring, 895
- imap_parse_dataitems, 895
- imap_parse_literal, 896
- imap_parse_nstring, 896
- imap_parse_qstring, 896
- imap_process, 897
- imap_range_build, 897
- imap_rename, 897
- imap_requeue, 897
- imap_search, 898
- imap_search_flag, 898
- imap_search_messages, 898
- imap_search_messages_body, 898
- imap_search_messages_date, 898
- imap_search_messages_date_compare, 899
- imap_search_messages_header, 899
- imap_search_messages_inner, 899
- imap_search_messages_range, 899
- imap_search_messages_size, 899
- imap_search_messages_text, 899
- IMAP_SEARCH_RECURSION_LIMIT, 879
- imap_select, 899
- imap_session_destroy, 900
- imap_session_update, 900
- imap_sort, 900
- imap_starttls, 900
- imap_status, 901
- imap_store, 901
- imap_subscribe, 901
- imap_unsubscribe, 901
- imap_update_flags, 901
- imap_valid_folder_name, 901
- imap_valid_sequence, 902
- servers/imap/messages.c
 - imap_append_message, 1090
 - imap_message_copier, 1090
 - imap_message_expunge, 1090
- servers/imap/parse.c
 - imap_command_log_safe, 2088
 - imap_command_parser, 2088
 - imap_get_ar_ar, 2089
 - imap_get_ptr, 2089
 - imap_get_st_ar, 2089
 - imap_get_type_ar, 2090
 - imap_parse_arguments, 2090
 - imap_parse_array, 2091
 - imap_parse_astring, 2091
 - imap_parse_literal, 2092
 - imap_parse_nstring, 2092
 - imap_parse_qstring, 2092
- servers/imap/search.c
 - imap_search_flag, 219
 - imap_search_messages, 219
 - imap_search_messages_body, 219
 - imap_search_messages_date, 220
 - imap_search_messages_date_compare, 220
 - imap_search_messages_header, 220
 - imap_search_messages_inner, 220
 - imap_search_messages_range, 220
 - imap_search_messages_size, 220
 - imap_search_messages_text, 220
 - MONTH_LOOKUP, 221
- servers/imap/sessions.c
 - imap_session_destroy, 1239
 - imap_session_update, 1239
- servers/molten/commands.c
 - molten_compare, 1618
 - molten_parse, 1618
 - molten_sort, 1618
- servers/molten/commands.h
 - molten_commands, 1634
- servers/molten/sessions.c
 - molten_session_destroy, 1240
- servers/pop/commands.c
 - pop_compare, 1619
 - pop_process, 1619
 - pop_requeue, 1619
 - pop_sort, 1619
- servers/pop/commands.h
 - pop_commands, 1635
- servers/pop/parse.c
 - pop_num_parse, 2094
 - pop_pass_parse, 2094
 - pop_top_parse, 2095
 - pop_user_parse, 2095
- servers/pop/pop.h
 - pop_capa, 957
 - pop_compare, 958
 - pop_delete, 958
 - pop_get_last, 958
 - pop_get_message, 958
 - pop_init, 959
 - pop_invalid, 959
 - pop_last, 959
 - pop_list, 960
 - pop_noop, 960
 - pop_num_parse, 960
 - pop_pass, 961
 - pop_pass_parse, 961
 - pop_process, 961
 - pop_quit, 962

- pop_requeue, 962
- pop_retr, 962
- pop_rset, 962
- pop_session_destroy, 963
- pop_session_reset, 963
- pop_sort, 963
- pop_starttls, 963
- pop_stat, 964
- pop_top, 964
- pop_top_parse, 964
- pop_total_messages, 965
- pop_total_size, 965
- pop_uidl, 965
- pop_user, 966
- pop_user_parse, 966
- servers/pop/sessions.c
 - pop_session_destroy, 1241
 - pop_session_reset, 1241
- servers/servers.h
 - M_SSL_BIO_NOCLOSE, 759
- servers/smtp/commands.c
 - smtp_compare, 1620
 - smtp_process, 1620
 - smtp_requeue, 1620
 - smtp_sort, 1620
- servers/smtp/commands.h
 - smtp_commands, 1636
- servers/smtp/datatier.c
 - smtp_check_authorized_from, 708
 - smtp_check_receive_quota, 709
 - smtp_check_transmit_quota, 709
 - smtp_fetch_authorization, 709
 - smtp_fetch_autoreply, 710
 - smtp_fetch_inbound, 710
 - smtp_fetch_rollmessages, 712
 - smtp_get_action, 712
 - smtp_insert_spamsig, 712
 - smtp_update_receive_stats, 713
 - smtp_update_transmission_stats, 713
- servers/smtp/parse.c
 - smtp_parse_auth, 2096
 - smtp_parse_helo_domain, 2096
 - smtp_parse_mail_from_path, 2097
 - smtp_parse_rcpt_to, 2097
- servers/smtp/relay.c
 - smtp_client_close, 739
 - smtp_client_connect, 739
 - smtp_client_send_data, 740
 - smtp_client_send_helo, 740
 - smtp_client_send_mailfrom, 740
 - smtp_client_send_nullfrom, 741
 - smtp_client_send_rcptto, 741
- servers/smtp/smtp.h
 - smtp_accept_message, 986
 - smtp_add_bypass_entry, 986
 - smtp_add_inbound, 987
 - smtp_add_outbound, 987
 - smtp_add_recipient, 987
 - smtp_auth_login, 988
 - smtp_auth_plain, 988
 - smtp_bounce, 988
 - smtp_bypass_check, 988
 - smtp_check_authorized_from, 989
 - smtp_check_duplicate_recipient, 989
 - smtp_check_filters, 989
 - smtp_check_greylist, 990
 - smtp_check_rbl, 990
 - smtp_check_receive_quota, 990
 - smtp_check_transmit_quota, 991
 - smtp_client_close, 991
 - smtp_client_connect, 992
 - smtp_client_send_data, 992
 - smtp_client_send_helo, 992
 - smtp_client_send_mailfrom, 993
 - smtp_client_send_nullfrom, 993
 - smtp_client_send_rcptto, 993
 - smtp_compare, 994
 - smtp_data, 994
 - smtp_disabled, 994
 - smtp_ehlo, 994
 - smtp_fetch_authorization, 995
 - smtp_fetch_autoreply, 995
 - smtp_fetch_inbound, 996
 - smtp_fetch_rollmessages, 997
 - smtp_forward_message, 997
 - smtp_free_inbound, 997
 - smtp_free_outbound, 998
 - smtp_free_recipients, 998
 - smtp_get_action, 998
 - smtp_helo, 999
 - smtp_init, 999
 - smtp_insert_spamsig, 999
 - smtp_invalid, 1000
 - smtp_list_free_filter, 1000
 - smtp_mail_from, 1000
 - smtp_noop, 1001
 - smtp_parse_auth, 1001
 - smtp_parse_helo_domain, 1001
 - smtp_parse_mail_from_path, 1002
 - smtp_parse_rcpt_to, 1002
 - smtp_process, 1002
 - smtp_quit, 1003
 - smtp_rcpt_to, 1003
 - smtp_relay_message, 1003
 - smtp_reply, 1004
 - smtp_requeue, 1004
 - smtp_rollout, 1004
 - smtp_rset, 1004
 - smtp_send_message, 1005
 - smtp_session_destroy, 1005
 - smtp_session_reset, 1005
 - smtp_sort, 1006
 - smtp_starttls, 1006
 - smtp_store_message, 1006

- smtp_store_spamsig, 1007
- smtp_update_receive_stats, 1007
- smtp_update_transmission_stats, 1007
- submission_init, 1008
- servers_alloc
 - engine/config/servers/servers.h, 753
 - servers.c, 747
- servers_config
 - engine/config/servers/servers.h, 753
 - servers.c, 747
- servers_encryption_start
 - engine/config/servers/servers.h, 754
 - servers.c, 747
- servers_encryption_stop
 - engine/config/servers/servers.h, 754
 - servers.c, 747
- servers_free
 - engine/config/servers/servers.h, 754
 - servers.c, 748
- servers_get_by_protocol
 - engine/config/servers/servers.h, 755
 - servers.c, 748
- servers_get_by_socket
 - engine/config/servers/servers.h, 755
 - servers.c, 748
- servers_get_count_using_port
 - engine/config/servers/servers.h, 755
 - servers.c, 749
- servers_network_start
 - engine/config/servers/servers.h, 756
 - servers.c, 749
- servers_network_stop
 - engine/config/servers/servers.h, 756
 - servers.c, 749
- servers_output_help
 - engine/config/servers/servers.h, 756
 - servers.c, 750
- servers_output_settings
 - engine/config/servers/servers.h, 756
 - servers.c, 750
- servers_set_value
 - engine/config/servers/servers.h, 757
 - servers.c, 750
- servers_validate
 - engine/config/servers/servers.h, 757
 - servers.c, 751
- sess_create
 - objects/sessions/sessions.c, 1230
 - objects/sessions/sessions.h, 974
- sess_destroy
 - objects/sessions/sessions.c, 1230
 - objects/sessions/sessions.h, 974
- sess_get
 - objects/sessions/sessions.c, 1231
 - objects/sessions/sessions.h, 974
- sess_key
 - objects/sessions/sessions.c, 1231
- objects/sessions/sessions.h, 975
- sess_number
 - objects/sessions/sessions.c, 1231
 - objects/sessions/sessions.h, 975
- sess_ref_add
 - objects/sessions/sessions.c, 1232
 - objects/sessions/sessions.h, 975
- sess_ref_dec
 - objects/sessions/sessions.c, 1232
 - objects/sessions/sessions.h, 976
- sess_ref_stamp
 - objects/sessions/sessions.c, 1232
 - objects/sessions/sessions.h, 976
- sess_ref_total
 - objects/sessions/sessions.c, 1232
 - objects/sessions/sessions.h, 976
- sess_refresh_check
 - objects/sessions/sessions.c, 1233
 - objects/sessions/sessions.h, 977
- sess_refresh_flush
 - objects/sessions/sessions.c, 1233
 - objects/sessions/sessions.h, 977
- sess_refresh_stamp
 - objects/sessions/sessions.c, 1233
 - objects/sessions/sessions.h, 977
- sess_release
 - objects/sessions/sessions.c, 1234
 - objects/sessions/sessions.h, 977
- sess_release_attachment
 - objects/sessions/sessions.c, 1234
 - objects/sessions/sessions.h, 978
- sess_release_composition
 - objects/sessions/sessions.c, 1234
 - objects/sessions/sessions.h, 978
- sess_serial_check
 - objects/sessions/sessions.c, 1235
 - objects/sessions/sessions.h, 978
- sess_token
 - objects/sessions/sessions.c, 1235
 - objects/sessions/sessions.h, 979
- sess_trigger
 - objects/sessions/sessions.c, 1235
 - objects/sessions/sessions.h, 979
- sess_update
 - objects/sessions/sessions.c, 1236
 - objects/sessions/sessions.h, 979
- session
 - __attribute__, 19
- SESSION_STATE_AUTHENTICATED
 - network/sessions.h, 971
- SESSION_STATE_NEUTRAL
 - network/sessions.h, 971
- SESSION_STATE_TERMINATED
 - network/sessions.h, 971
- session_state
 - __attribute__, 19
- session_t

- network/sessions.h, 971
- session_timeout
 - magma_t, 99
- sessions
 - magma_t, 100
 - object_cache_t, 116
 - objects/sessions/sessions.c, 1236
- set
 - magma_keys_t, 87
- set_cache_location
 - cache_pub.c, 1710
- set_cache_permissions
 - cache_pub.c, 1710
- set_dbg_level
 - misc_pub.c, 1560
- sgnt_resolv_destroy_dmtplib_session
 - dmtplib_pub.c, 1727
- sgnt_resolv_dmtplib_connect
 - dmtplib_pub.c, 1727
- sgnt_resolv_dmtplib_data
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_ehlo
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_get_mode
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_get_sigmet
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_help
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_history
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_mail_from
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_noop
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_quit
 - dmtplib_pub.c, 1728
- sgnt_resolv_dmtplib_rcpt_to
 - dmtplib_pub.c, 1729
- sgnt_resolv_dmtplib_reset
 - dmtplib_pub.c, 1729
- sgnt_resolv_dmtplib_stats
 - dmtplib_pub.c, 1729
- sgnt_resolv_dmtplib_str_to_mode
 - dmtplib_pub.c, 1729
- sgnt_resolv_dmtplib_verify_sigmet
 - dmtplib_pub.c, 1729
- SHA1_Final_d
 - symbols.c, 2004
 - symbols.h, 2053
- SHA1_Init_d
 - symbols.c, 2004
 - symbols.h, 2053
- SHA1_Update_d
 - symbols.c, 2005
 - symbols.h, 2053
- SHA256_Final_d
 - symbols.c, 2005
 - symbols.h, 2053
- SHA256_Init_d
 - symbols.c, 2005
 - symbols.h, 2054
- SHA256_Update_d
 - symbols.h, 2054
- SHA512_d
 - symbols.h, 2054
- SHA512_Final_d
 - symbols.c, 2005
 - symbols.h, 2054
- SHA512_Init_d
 - symbols.c, 2005
 - symbols.h, 2054
- SHA512_Update_d
 - symbols.h, 2054
- SHA_160_SIZE
 - misc.h, 1552
- SHA_256_SIZE
 - misc.h, 1552
- SHA_512_B64_SIZE
 - misc.h, 1552
- SHA_512_SIZE
 - misc.h, 1552
- sha_databuf_t, 133
 - data, 133
 - len, 133
- shadow
 - cached_object, 33
- shortcuts.c
 - pl_char_get, 528
 - pl_clone, 529
 - pl_data_get, 529
 - pl_empty, 529
 - pl_inc, 529
 - pl_init, 530
 - pl_length_get, 530
 - pl_length_int, 531
 - pl_null, 531
 - pl_set, 531
 - pl_starts_with_char, 531
- sig
 - generated_data_t, 74
 - test_data_t, 159
- sig_hup_mutex
 - signal.c, 773
- signal.c
 - sig_hup_mutex, 773
 - signal_refresh, 771
 - signal_segfault, 771
 - signal_shutdown, 772
 - signal_start, 772
 - signal_status, 772
 - signal_thread_start, 773
- signal_name
 - host.h, 301

- signals.c, 318
- signal_refresh
 - signal.c, 771
- signal_segfault
 - context.h, 764
 - signal.c, 771
- signal_shutdown
 - context.h, 765
 - signal.c, 772
- signal_start
 - context.h, 765
 - signal.c, 772
- signal_status
 - context.h, 766
 - signal.c, 772
- signal_thread_start
 - context.h, 766
 - signal.c, 773
- signals.c
 - signal_name, 318
- signature
 - teacher_data_t, 157
- signature.c
 - signature_full_get, 1888
 - signature_full_verify, 1888
 - signature_tree_add, 1888
 - signature_tree_alloc, 1889
 - signature_tree_cleanup, 1889
 - signature_tree_free, 1889
 - signature_tree_get, 1889
 - signature_tree_verify, 1889
- signature_full_get
 - chunks.h, 1878
 - signature.c, 1888
- signature_full_verify
 - chunks.h, 1878
 - signature.c, 1888
- signature_tree_add
 - chunks.h, 1878
 - signature.c, 1888
- signature_tree_alloc
 - chunks.h, 1879
 - signature.c, 1889
- signature_tree_cleanup
 - chunks.h, 1879
 - signature.c, 1889
- signature_tree_free
 - chunks.h, 1879
 - signature.c, 1889
- signature_tree_get
 - chunks.h, 1879
 - signature.c, 1889
- signature_tree_verify
 - chunks.h, 1879
 - signature.c, 1889
- signatures
 - magma_t, 100
- signatures.c
 - mail_build_signature, 1165
 - mail_discover_encoding, 1166
 - mail_discover_insertion_point, 1166
 - mail_discover_type, 1167
 - mail_extract_tag, 1167
 - mail_get_boundary, 1167
 - mail_get_chunk, 1168
 - mail_insert_chunk_base64, 1168
 - mail_insert_chunk_text, 1169
 - mail_modify_part, 1169
 - mail_signature_add, 1170
- signet
 - magma_t, 100
 - signet_field_t, 136
 - smtp_inbound_prefs_t, 145
- signet-ssl.h
 - _deserialize_ocsp_response_cb, 1770
 - _destroy_ocsp_response_cb, 1770
 - _domain_wildcard_check, 1771
 - _dump_ocsp_response_cb, 1771
 - _get_cache_ocsp_id, 1771
 - _get_cert_store, 1772
 - _ocsp_response_callback, 1772
 - _serialize_ocsp_response_cb, 1772
 - _ssl_fd_loop, 1773
 - _validate_self_signed, 1773
 - _verify_certificate_callback, 1773
 - CA_DIR, 1770
 - CA_FILE, 1770
 - CRL_FILE, 1770
 - PUBLIC_FUNC_DECL, 1774
- signet.c
 - dime_sgnt_enckey_fetch, 1791
 - dime_sgnt_enckey_set, 1791
 - dime_sgnt_fid_count_get, 1791
 - dime_sgnt_fid_exists, 1791
 - dime_sgnt_fid_num_fetch, 1792
 - dime_sgnt_fid_num_remove, 1792
 - dime_sgnt_field_defined_create, 1792
 - dime_sgnt_field_defined_set, 1793
 - dime_sgnt_field_undefined_create, 1793
 - dime_sgnt_field_undefined_fetch, 1793
 - dime_sgnt_field_undefined_remove, 1794
 - dime_sgnt_file_create, 1794
 - dime_sgnt_fingerprint_crypto, 1794
 - dime_sgnt_fingerprint_full, 1795
 - dime_sgnt_fingerprint_id, 1795
 - dime_sgnt_fingerprint_ssr, 1795
 - dime_sgnt_id_fetch, 1796
 - dime_sgnt_id_set, 1796
 - dime_sgnt_msg_sig_verify, 1796
 - dime_sgnt_sig_coc_sign, 1797
 - dime_sgnt_sig_crypto_sign, 1797
 - dime_sgnt_sig_full_sign, 1797
 - dime_sgnt_sig_id_sign, 1797
 - dime_sgnt_sig_ssr_sign, 1798

- dime_sgnt_signet_b64_deserialize, 1798
- dime_sgnt_signet_b64_serialize, 1798
- dime_sgnt_signet_binary_deserialize, 1799
- dime_sgnt_signet_binary_serialize, 1799
- dime_sgnt_signet_create, 1799
- dime_sgnt_signet_create_w_keys, 1799
- dime_sgnt_signet_crypto_split, 1800
- dime_sgnt_signet_destroy, 1800
- dime_sgnt_signet_dump, 1800
- dime_sgnt_signet_dupe, 1800
- dime_sgnt_signet_full_split, 1801
- dime_sgnt_signet_load, 1801
- dime_sgnt_signkey_fetch, 1801
- dime_sgnt_signkey_set, 1801
- dime_sgnt_signkeys_msg_fetch, 1802
- dime_sgnt_signkeys_signet_fetch, 1802
- dime_sgnt_signkeys_software_fetch, 1802
- dime_sgnt_signkeys_tls_fetch, 1803
- dime_sgnt_sok_create, 1803
- dime_sgnt_sok_num_fetch, 1803
- dime_sgnt_state_to_str, 1804
- dime_sgnt_type_get, 1804
- dime_sgnt_type_set, 1804
- dime_sgnt_validate_all, 1805
- signet.h
 - dime_sgnt_enckey_fetch, 1809
 - dime_sgnt_enckey_set, 1810
 - dime_sgnt_fid_count_get, 1810
 - dime_sgnt_fid_exists, 1810
 - dime_sgnt_fid_num_fetch, 1810
 - dime_sgnt_fid_num_remove, 1811
 - dime_sgnt_field_defined_create, 1811
 - dime_sgnt_field_defined_set, 1811
 - dime_sgnt_field_undefined_create, 1812
 - dime_sgnt_field_undefined_fetch, 1812
 - dime_sgnt_field_undefined_remove, 1812
 - dime_sgnt_file_create, 1813
 - dime_sgnt_fingerprint_crypto, 1813
 - dime_sgnt_fingerprint_full, 1813
 - dime_sgnt_fingerprint_id, 1814
 - dime_sgnt_fingerprint_ssr, 1814
 - dime_sgnt_id_fetch, 1814
 - dime_sgnt_id_set, 1815
 - dime_sgnt_msg_sig_verify, 1815
 - dime_sgnt_sig_coc_sign, 1815
 - dime_sgnt_sig_crypto_sign, 1815
 - dime_sgnt_sig_full_sign, 1816
 - dime_sgnt_sig_id_sign, 1816
 - dime_sgnt_sig_ssr_sign, 1816
 - dime_sgnt_signet_b64_deserialize, 1817
 - dime_sgnt_signet_b64_serialize, 1817
 - dime_sgnt_signet_binary_deserialize, 1817
 - dime_sgnt_signet_binary_serialize, 1817
 - dime_sgnt_signet_create, 1818
 - dime_sgnt_signet_create_w_keys, 1818
 - dime_sgnt_signet_crypto_split, 1818
 - dime_sgnt_signet_destroy, 1819
 - dime_sgnt_signet_dump, 1819
 - dime_sgnt_signet_dupe, 1819
 - dime_sgnt_signet_full_split, 1819
 - dime_sgnt_signet_load, 1820
 - dime_sgnt_signkey_fetch, 1820
 - dime_sgnt_signkey_set, 1820
 - dime_sgnt_signkeys_msg_fetch, 1820
 - dime_sgnt_signkeys_signet_fetch, 1821
 - dime_sgnt_signkeys_software_fetch, 1821
 - dime_sgnt_signkeys_tls_fetch, 1821
 - dime_sgnt_sok_create, 1822
 - dime_sgnt_sok_num_fetch, 1822
 - dime_sgnt_state_to_str, 1822
 - dime_sgnt_type_get, 1823
 - dime_sgnt_type_set, 1823
 - dime_sgnt_validate_all, 1823
 - SIGNET_TYPE_ERROR, 1809
 - SIGNET_TYPE_ORG, 1809
 - SIGNET_TYPE_SSR, 1809
 - SIGNET_TYPE_USER, 1809
 - signet_state_t, 1809
 - signet_type_t, 1809
 - SS_BROKEN_COC, 1809
 - SS_CRYPTO, 1809
 - SS_FULL, 1809
 - SS_ID, 1809
 - SS_INCOMPLETE, 1809
 - SS_INVALID, 1809
 - SS_MALFORMED, 1809
 - SS_OVERFLOW, 1809
 - SS_SSR, 1809
 - SS_UNKNOWN, 1809
- signet/common.h
 - B64, 1576
 - DIME_ENCRYPTED_MSG, 1576
 - DIME_ENCRYPTED_ORG_KEYS, 1576
 - DIME_ENCRYPTED_USER_KEYS, 1576
 - DIME_MSG_TRACING, 1576
 - DIME_ORG_KEYS, 1576
 - DIME_ORG_SIGNET, 1576
 - DIME_SSR, 1576
 - DIME_USER_KEYS, 1576
 - DIME_USER_SIGNET, 1576
 - DIME_NUMBER_SIZE, 1574
 - dime_number_t, 1576
 - dime_number_to_str, 1579
 - field_data_t, 1576
 - FIELD_NAME_MAX_SIZE, 1574
 - HEX, 1576
 - KEYS_ORG_PRIVATE_ENC, 1576
 - KEYS_ORG_PRIVATE_POK, 1576
 - KEYS_ORG_PRIVATE_SOK, 1576
 - KEYS_USER_PRIVATE_ENC, 1576
 - KEYS_USER_PRIVATE_SIGN, 1576
 - KEYS_FID_MAX, 1574
 - KEYS_HEADER_SIZE, 1574
 - keys_org_t, 1576

keys_user_t, [1576](#)
 PNG, [1576](#)
 SIGNET_ORG_ABUSE, [1577](#)
 SIGNET_ORG_ADDRESS, [1577](#)
 SIGNET_ORG_ADMIN, [1577](#)
 SIGNET_ORG_COUNTRY, [1577](#)
 SIGNET_ORG_CRYPTOSIG, [1577](#)
 SIGNET_ORG_CRYPTOCURRENCY, [1577](#)
 SIGNET_ORG_CURRENCY, [1577](#)
 SIGNET_ORG_ENC_KEY, [1577](#)
 SIGNET_ORG_EXTENSIONS, [1577](#)
 SIGNET_ORG_FULL_SIG, [1577](#)
 SIGNET_ORG_ID, [1577](#)
 SIGNET_ORG_ID_SIG, [1577](#)
 SIGNET_ORG_LANGUAGE, [1577](#)
 SIGNET_ORG_MAIL_CERT, [1577](#)
 SIGNET_ORG_MAIL_HOST, [1577](#)
 SIGNET_ORG_MOTTO, [1577](#)
 SIGNET_ORG_MSG_SIZE_LIM, [1577](#)
 SIGNET_ORG_NAME, [1577](#)
 SIGNET_ORG_ONION_ACCESS_CERT, [1577](#)
 SIGNET_ORG_ONION_ACCESS_HOST, [1577](#)
 SIGNET_ORG_ONION_DELIVERY_CERT, [1577](#)
 SIGNET_ORG_ONION_DELIVERY_HOST, [1577](#)
 SIGNET_ORG_PHONE, [1577](#)
 SIGNET_ORG_PHOTO, [1577](#)
 SIGNET_ORG_POK, [1577](#)
 SIGNET_ORG_POSTAL, [1577](#)
 SIGNET_ORG_PROVINCE, [1577](#)
 SIGNET_ORG_SOK, [1577](#)
 SIGNET_ORG_SUPPORT, [1577](#)
 SIGNET_ORG_UNDEFINED, [1577](#)
 SIGNET_ORG_WEB_CERT, [1577](#)
 SIGNET_ORG_WEB_HOST, [1577](#)
 SIGNET_ORG_WEB_LOCATION, [1577](#)
 SIGNET_ORG_WEBSITE, [1577](#)
 SIGNET_SOK_MSG, [1579](#)
 SIGNET_SOK_NONE, [1579](#)
 SIGNET_SOK_SIGNET, [1579](#)
 SIGNET_SOK_SOFTWARE, [1579](#)
 SIGNET_SOK_TLS, [1579](#)
 SIGNET_SSR_ALT_KEY, [1577](#)
 SIGNET_SSR_COC_SIG, [1577](#)
 SIGNET_SSR_ENC_KEY, [1577](#)
 SIGNET_SSR_SIGN_KEY, [1577](#)
 SIGNET_SSR_SSR_SIG, [1577](#)
 SIGNET_USER_ADDRESS, [1578](#)
 SIGNET_USER_ALMA_MATER, [1578](#)
 SIGNET_USER_ALT_KEY, [1578](#)
 SIGNET_USER_ALTERNATE_ADDRESS, [1578](#)
 SIGNET_USER_COC_SIG, [1578](#)
 SIGNET_USER_CODECS, [1578](#)
 SIGNET_USER_COUNTRY, [1578](#)
 SIGNET_USER_CRYPTOSIG, [1578](#)
 SIGNET_USER_CRYPTOCURRENCY, [1578](#)
 SIGNET_USER_CURRENCY, [1578](#)
 SIGNET_USER_EMPLOYER, [1578](#)
 SIGNET_USER_ENC_KEY, [1578](#)
 SIGNET_USER_ENDORSEMENTS, [1578](#)
 SIGNET_USER_EXTENSIONS, [1578](#)
 SIGNET_USER_FULL_SIG, [1578](#)
 SIGNET_USER_GENDER, [1578](#)
 SIGNET_USER_ID, [1578](#)
 SIGNET_USER_ID_SIG, [1578](#)
 SIGNET_USER_LANGUAGE, [1578](#)
 SIGNET_USER_MOTTO, [1578](#)
 SIGNET_USER_MSG_SIZE_LIM, [1578](#)
 SIGNET_USER_NAME, [1578](#)
 SIGNET_USER_PHONE, [1578](#)
 SIGNET_USER_PHOTO, [1578](#)
 SIGNET_USER_POLITICAL_PARTY, [1578](#)
 SIGNET_USER_POSTAL, [1578](#)
 SIGNET_USER_PROVINCE, [1578](#)
 SIGNET_USER_RESUME, [1578](#)
 SIGNET_USER_SIGN_KEY, [1578](#)
 SIGNET_USER_SSR_SIG, [1578](#)
 SIGNET_USER_SUPERVISOR, [1578](#)
 SIGNET_USER_TITLE, [1578](#)
 SIGNET_USER_UNDEFINED, [1578](#)
 SIGNET_FID_MAX, [1574](#)
 SIGNET_HEADER_SIZE, [1575](#)
 SIGNET_KEY_ORG, [1575](#)
 SIGNET_KEY_USER, [1575](#)
 SIGNET_MAX_SIZE, [1575](#)
 SIGNET_ORG, [1575](#)
 signet_org_field_keys, [1579](#)
 signet_org_field_t, [1576](#)
 signet_ssr_field_keys, [1579](#)
 signet_ssr_field_t, [1577](#)
 SIGNET_USER, [1575](#)
 signet_user_field_keys, [1579](#)
 signet_user_field_t, [1577](#)
 SIGNET_VER_NO, [1575](#)
 SIGNKEY_DEFAULT_FORMAT, [1578](#)
 signkey_format_t, [1578](#)
 sok_permissions_t, [1578](#)
 UNICODE, [1576](#)
 UNSIGNED_MAX_1_BYTE, [1575](#)
 UNSIGNED_MAX_2_BYTE, [1575](#)
 UNSIGNED_MAX_3_BYTE, [1575](#)
 SIGNET_ORG_ABUSE
 signet/common.h, [1577](#)
 SIGNET_ORG_ADDRESS
 signet/common.h, [1577](#)
 SIGNET_ORG_ADMIN
 signet/common.h, [1577](#)
 SIGNET_ORG_COUNTRY
 signet/common.h, [1577](#)
 SIGNET_ORG_CRYPTOSIG
 signet/common.h, [1577](#)
 SIGNET_ORG_CRYPTOCURRENCY
 signet/common.h, [1577](#)
 SIGNET_ORG_CURRENCY
 signet/common.h, [1577](#)

SIGNET_ORG_ENC_KEY
 [signet/common.h, 1577](#)

SIGNET_ORG_EXTENSIONS
 [signet/common.h, 1577](#)

SIGNET_ORG_FULL_SIG
 [signet/common.h, 1577](#)

SIGNET_ORG_ID
 [signet/common.h, 1577](#)

SIGNET_ORG_ID_SIG
 [signet/common.h, 1577](#)

SIGNET_ORG_LANGUAGE
 [signet/common.h, 1577](#)

SIGNET_ORG_MAIL_CERT
 [signet/common.h, 1577](#)

SIGNET_ORG_MAIL_HOST
 [signet/common.h, 1577](#)

SIGNET_ORG_MOTTO
 [signet/common.h, 1577](#)

SIGNET_ORG_MSG_SIZE_LIM
 [signet/common.h, 1577](#)

SIGNET_ORG_NAME
 [signet/common.h, 1577](#)

SIGNET_ORG_ONION_ACCESS_CERT
 [signet/common.h, 1577](#)

SIGNET_ORG_ONION_ACCESS_HOST
 [signet/common.h, 1577](#)

SIGNET_ORG_ONION_DELIVERY_CERT
 [signet/common.h, 1577](#)

SIGNET_ORG_ONION_DELIVERY_HOST
 [signet/common.h, 1577](#)

SIGNET_ORG_PHONE
 [signet/common.h, 1577](#)

SIGNET_ORG_PHOTO
 [signet/common.h, 1577](#)

SIGNET_ORG_POK
 [signet/common.h, 1577](#)

SIGNET_ORG_POSTAL
 [signet/common.h, 1577](#)

SIGNET_ORG_PROVINCE
 [signet/common.h, 1577](#)

SIGNET_ORG_SOK
 [signet/common.h, 1577](#)

SIGNET_ORG_SUPPORT
 [signet/common.h, 1577](#)

SIGNET_ORG_UNDEFINED
 [signet/common.h, 1577](#)

SIGNET_ORG_WEB_CERT
 [signet/common.h, 1577](#)

SIGNET_ORG_WEB_HOST
 [signet/common.h, 1577](#)

SIGNET_ORG_WEB_LOCATION
 [signet/common.h, 1577](#)

SIGNET_ORG_WEBSITE
 [signet/common.h, 1577](#)

SIGNET_SOK_MSG
 [signet/common.h, 1579](#)

SIGNET_SOK_NONE
 [signet/common.h, 1579](#)

SIGNET_SOK_SIGNET
 [signet/common.h, 1579](#)

SIGNET_SOK_SOFTWARE
 [signet/common.h, 1579](#)

SIGNET_SOK_TLS
 [signet/common.h, 1579](#)

SIGNET_SSR_ALT_KEY
 [signet/common.h, 1577](#)

SIGNET_SSR_COC_SIG
 [signet/common.h, 1577](#)

SIGNET_SSR_ENC_KEY
 [signet/common.h, 1577](#)

SIGNET_SSR_SIGN_KEY
 [signet/common.h, 1577](#)

SIGNET_SSR_SSR_SIG
 [signet/common.h, 1577](#)

SIGNET_TYPE_ERROR
 [signet.h, 1809](#)

SIGNET_TYPE_ORG
 [signet.h, 1809](#)

SIGNET_TYPE_SSR
 [signet.h, 1809](#)

SIGNET_TYPE_USER
 [signet.h, 1809](#)

SIGNET_USER_ADDRESS
 [signet/common.h, 1578](#)

SIGNET_USER_ALMA_MATER
 [signet/common.h, 1578](#)

SIGNET_USER_ALT_KEY
 [signet/common.h, 1578](#)

SIGNET_USER_ALTERNATE_ADDRESS
 [signet/common.h, 1578](#)

SIGNET_USER_COC_SIG
 [signet/common.h, 1578](#)

SIGNET_USER_CODECS
 [signet/common.h, 1578](#)

SIGNET_USER_COUNTRY
 [signet/common.h, 1578](#)

SIGNET_USER_CRYPTOSIG
 [signet/common.h, 1578](#)

SIGNET_USER_CRYPTOCURRENCY
 [signet/common.h, 1578](#)

SIGNET_USER_CURRENCY
 [signet/common.h, 1578](#)

SIGNET_USER_EMPLOYER
 [signet/common.h, 1578](#)

SIGNET_USER_ENC_KEY
 [signet/common.h, 1578](#)

SIGNET_USER_ENDORSEMENTS
 [signet/common.h, 1578](#)

SIGNET_USER_EXTENSIONS
 [signet/common.h, 1578](#)

SIGNET_USER_FULL_SIG
 [signet/common.h, 1578](#)

SIGNET_USER_GENDER
 [signet/common.h, 1578](#)

- SIGNET_USER_ID
 - signet/common.h, 1578
- SIGNET_USER_ID_SIG
 - signet/common.h, 1578
- SIGNET_USER_LANGUAGE
 - signet/common.h, 1578
- SIGNET_USER_MOTTO
 - signet/common.h, 1578
- SIGNET_USER_MSG_SIZE_LIM
 - signet/common.h, 1578
- SIGNET_USER_NAME
 - signet/common.h, 1578
- SIGNET_USER_PHONE
 - signet/common.h, 1578
- SIGNET_USER_PHOTO
 - signet/common.h, 1578
- SIGNET_USER_POLITICAL_PARTY
 - signet/common.h, 1578
- SIGNET_USER_POSTAL
 - signet/common.h, 1578
- SIGNET_USER_PROVINCE
 - signet/common.h, 1578
- SIGNET_USER_RESUME
 - signet/common.h, 1578
- SIGNET_USER_SIGN_KEY
 - signet/common.h, 1578
- SIGNET_USER_SSR_SIG
 - signet/common.h, 1578
- SIGNET_USER_SUPERVISOR
 - signet/common.h, 1578
- SIGNET_USER_TITLE
 - signet/common.h, 1578
- SIGNET_USER_UNDEFINED
 - signet/common.h, 1578
- signet_author
 - dmime_object_t, 49
- signet_destination
 - dmime_object_t, 49
- SIGNET_FID_MAX
 - signet/common.h, 1574
- signet_field_key_t, 134
 - bytes_data_size, 134
 - bytes_name_size, 134
 - data_size, 134
 - data_type, 134
 - description, 134
 - name, 134
 - required, 135
 - unique, 135
- signet_field_t, 136
 - data_offset, 136
 - data_size, 136
 - id_offset, 136
 - key, 136
 - name_offset, 136
 - name_size, 136
 - next, 136
 - signet, 136
- SIGNET_HEADER_SIZE
 - signet/common.h, 1575
- SIGNET_KEY_ORG
 - signet/common.h, 1575
- SIGNET_KEY_USER
 - signet/common.h, 1575
- SIGNET_MAX_SIZE
 - signet/common.h, 1575
- SIGNET_ORG
 - signet/common.h, 1575
- signet_org_field_keys
 - general.c, 1787
 - signet/common.h, 1579
- signet_org_field_t
 - signet/common.h, 1576
- signet_origin
 - dmime_object_t, 49
- signet_recipient
 - dmime_object_t, 49
- signet_ssr_field_keys
 - general.c, 1787
 - signet/common.h, 1579
- signet_ssr_field_t
 - signet/common.h, 1577
- signet_state_t
 - signet.h, 1809
- signet_t, 138
 - data, 138
 - fields, 138
 - size, 138
 - type, 138
- signet_type_t
 - signet.h, 1809
- SIGNET_USER
 - signet/common.h, 1575
- signet_user_field_keys
 - general.c, 1787
 - signet/common.h, 1579
- signet_user_field_t
 - signet/common.h, 1577
- SIGNET_VER_NO
 - signet/common.h, 1575
- signets.h
 - org_signet_alloc, 1915
 - org_signet_fingerprint, 1915
 - org_signet_free, 1916
 - org_signet_generate, 1916
 - org_signet_get, 1916
 - org_signet_length, 1916
 - org_signet_set, 1916
 - org_signet_verify, 1916
 - user_request_generate, 1917
 - user_request_get, 1917
 - user_request_length, 1917
 - user_request_rotation, 1917
 - user_request_set, 1917

- user_request_sign, 1917
- user_request_verify_chain_of_custody, 1918
- user_request_verify_self, 1918
- user_signet_alloc, 1918
- user_signet_fingerprint, 1918
- user_signet_free, 1918
- user_signet_get, 1918
- user_signet_length, 1919
- user_signet_set, 1919
- user_signet_verify_chain_of_custody, 1919
- user_signet_verify_org, 1919
- user_signet_verify_self, 1919
- signets/orgs.c
 - org_signet_alloc, 1865
 - org_signet_fingerprint, 1865
 - org_signet_free, 1865
 - org_signet_generate, 1865
 - org_signet_get, 1866
 - org_signet_length, 1866
 - org_signet_set, 1866
 - org_signet_verify, 1866
- signets/users.c
 - user_signet_alloc, 1869
 - user_signet_fingerprint, 1869
 - user_signet_free, 1869
 - user_signet_get, 1869
 - user_signet_length, 1869
 - user_signet_set, 1870
 - user_signet_verify_chain_of_custody, 1870
 - user_signet_verify_org, 1870
 - user_signet_verify_self, 1870
- SIGNKEY_DEFAULT_FORMAT
 - signet/common.h, 1578
- signkey_format_t
 - signet/common.h, 1578
- signkeys
 - dnskey, 57
 - ds, 60
- signum
 - smtp_inbound_prefs_t, 145
 - teacher_data_t, 157
- SIGUNUSED
 - core.h, 228
- size
 - batch_heap_t, 28
 - dmime_message_t, 47
 - dmtp_argument_t, 51
 - signet_t, 138
- size_conv_bl
 - numbers.c, 414
 - numbers.h, 432
- sk
 - random_data_t, 122
 - test_data_t, 159
- sk_num_d
 - symbols.c, 2005
 - symbols.h, 2054
- sk_pop_d
 - symbols.c, 2005
 - symbols.h, 2054
- sk_pop_free_d
 - symbols.c, 2005
 - symbols.h, 2054
- sk_value_d
 - symbols.c, 2005
 - symbols.h, 2054
- SKEY_EMPTY
 - general.c, 1786
- SKEY_SIZE1
 - general.c, 1786
- SKEY_SIZE2
 - general.c, 1786
- slab
 - secure.c, 391
- slots.c
 - PRIME_ACTOR_AUTHOR, 1890
 - PRIME_ACTOR_DESTINATION, 1890
 - PRIME_ACTOR_NONE, 1890
 - PRIME_ACTOR_ORIGIN, 1890
 - PRIME_ACTOR_RECIPIENT, 1891
 - slots_actors, 1891
 - slots_alloc, 1891
 - slots_buffer, 1891
 - slots_cleanup, 1891
 - slots_count, 1891
 - slots_free, 1892
 - slots_get, 1892
 - slots_key, 1892
 - slots_set, 1892
- slots_actors
 - chunks.h, 1879
 - slots.c, 1891
- slots_alloc
 - chunks.h, 1880
 - slots.c, 1891
- slots_buffer
 - chunks.h, 1880
 - slots.c, 1891
- slots_cleanup
 - chunks.h, 1880
 - slots.c, 1891
- slots_count
 - chunks.h, 1880
 - slots.c, 1891
- slots_free
 - chunks.h, 1880
 - slots.c, 1892
- slots_get
 - chunks.h, 1880
 - slots.c, 1892
- slots_key
 - chunks.h, 1881
 - slots.c, 1892
- slots_set

- chunks.h, 1881
- slots.c, 1892
- SMTP
 - engine/config/servers/servers.h, 753
- smtp
 - magma_t, 100
- smtp.c
 - smtp_auth_login, 2151
 - smtp_auth_plain, 2151
 - smtp_data, 2151
 - smtp_data_finish, 2151
 - smtp_data_inbound, 2151
 - smtp_data_outbound, 2151
 - smtp_data_read, 2152
 - smtp_disabled, 2152
 - smtp_ehlo, 2152
 - smtp_helo, 2152
 - smtp_init, 2153
 - smtp_invalid, 2153
 - smtp_mail_from, 2153
 - smtp_noop, 2154
 - smtp_quit, 2154
 - smtp_rcpt_to, 2154
 - smtp_rset, 2155
 - smtp_starttls, 2155
 - submission_init, 2155
- smtp/session.c
 - smtp_add_inbound, 2101
 - smtp_add_outbound, 2101
 - smtp_add_recipient, 2102
 - smtp_check_duplicate_recipient, 2102
 - smtp_free_inbound, 2102
 - smtp_free_outbound, 2103
 - smtp_free_recipients, 2103
 - smtp_list_free_filter, 2103
 - smtp_session_destroy, 2103
 - smtp_session_reset, 2104
- SMTP_ACTION_BOUNCE
 - network/smtp.h, 981
- SMTP_ACTION_DELETE
 - network/smtp.h, 981
- SMTP_ACTION_ERROR
 - network/smtp.h, 981
- SMTP_ACTION_MARK
 - network/smtp.h, 981
- SMTP_ACTION_MARK_READ
 - network/smtp.h, 981
- SMTP_ACTION_REJECT
 - network/smtp.h, 981
- SMTP_ACTION_UNDEFINED
 - network/smtp.h, 981
- SMTP_FILTER_ACTION_DELETE
 - network/smtp.h, 982
- SMTP_FILTER_ACTION_LABEL
 - network/smtp.h, 982
- SMTP_FILTER_ACTION_MARK_READ
 - network/smtp.h, 982

- SMTP_FILTER_ACTION_MOVE
 - network/smtp.h, 982
- SMTP_FILTER_ACTION_NONE
 - network/smtp.h, 982
- SMTP_FILTER_LOCATION_BODY
 - network/smtp.h, 982
- SMTP_FILTER_LOCATION_ENTIRE
 - network/smtp.h, 982
- SMTP_FILTER_LOCATION_FIELD
 - network/smtp.h, 982
- SMTP_FILTER_LOCATION_HEADER
 - network/smtp.h, 982
- SMTP_FILTER_TYPE_CONTAINS
 - network/smtp.h, 982
- SMTP_FILTER_TYPE_ENDS
 - network/smtp.h, 982
- SMTP_FILTER_TYPE_EXACT
 - network/smtp.h, 982
- SMTP_FILTER_TYPE_REGEX
 - network/smtp.h, 982
- SMTP_FILTER_TYPE_STARTS
 - network/smtp.h, 982
- SMTP_MARK_FILTERED
 - network/smtp.h, 982
- SMTP_MARK_NONE
 - network/smtp.h, 981
- SMTP_MARK_PHISH
 - network/smtp.h, 981
- SMTP_MARK_RBL
 - network/smtp.h, 981
- SMTP_MARK_READ
 - network/smtp.h, 981
- SMTP_MARK_SPAM
 - network/smtp.h, 981
- SMTP_MARK_SPOOF
 - network/smtp.h, 982
- SMTP_MARK_VIRUS
 - network/smtp.h, 981
- SMTP_OUTCOME_BOUNCE_DKIM
 - network/smtp.h, 982
- SMTP_OUTCOME_BOUNCE_PHISH
 - network/smtp.h, 982
- SMTP_OUTCOME_BOUNCE_RBL
 - network/smtp.h, 982
- SMTP_OUTCOME_BOUNCE_SPAM
 - network/smtp.h, 982
- SMTP_OUTCOME_BOUNCE_SPF
 - network/smtp.h, 982
- SMTP_OUTCOME_BOUNCE_VIRUS
 - network/smtp.h, 982
- SMTP_OUTCOME_PERM_FAILURE
 - network/smtp.h, 982
- SMTP_OUTCOME_SUCESS
 - network/smtp.h, 982
- SMTP_OUTCOME_TEMP_LOCKED
 - network/smtp.h, 982
- SMTP_OUTCOME_TEMP_OVERQUOTA

- network/smtp.h, 982
- SMTP_OUTCOME_TEMP_SERVER
 - network/smtp.h, 982
- smtp_accept_message
 - accept.c, 2145
 - servers/smtp/smtp.h, 986
- smtp_add_bypass_entry
 - checkers.c, 2147
 - servers/smtp/smtp.h, 986
- smtp_add_inbound
 - servers/smtp/smtp.h, 987
 - smtp/session.c, 2101
- smtp_add_outbound
 - servers/smtp/smtp.h, 987
 - smtp/session.c, 2101
- smtp_add_recipient
 - servers/smtp/smtp.h, 987
 - smtp/session.c, 2102
- smtp_auth_login
 - servers/smtp/smtp.h, 988
 - smtp.c, 2151
- smtp_auth_plain
 - servers/smtp/smtp.h, 988
 - smtp.c, 2151
- smtp_bounce
 - servers/smtp/smtp.h, 988
 - transmit.c, 2156
- smtp_bypass_check
 - checkers.c, 2147
 - servers/smtp/smtp.h, 988
- smtp_check_authorized_from
 - servers/smtp/datatier.c, 708
 - servers/smtp/smtp.h, 989
- smtp_check_duplicate_recipient
 - servers/smtp/smtp.h, 989
 - smtp/session.c, 2102
- smtp_check_filters
 - checkers.c, 2148
 - servers/smtp/smtp.h, 989
- smtp_check_greylist
 - checkers.c, 2148
 - servers/smtp/smtp.h, 990
- smtp_check_rbl
 - checkers.c, 2148
 - servers/smtp/smtp.h, 990
- smtp_check_receive_quota
 - servers/smtp/datatier.c, 709
 - servers/smtp/smtp.h, 990
- smtp_check_transmit_quota
 - servers/smtp/datatier.c, 709
 - servers/smtp/smtp.h, 991
- smtp_client_close
 - servers/smtp/relay.c, 739
 - servers/smtp/smtp.h, 991
- smtp_client_connect
 - servers/smtp/relay.c, 739
 - servers/smtp/smtp.h, 992
- smtp_client_send_data
 - servers/smtp/relay.c, 740
 - servers/smtp/smtp.h, 992
- smtp_client_send_helo
 - servers/smtp/relay.c, 740
 - servers/smtp/smtp.h, 992
- smtp_client_send_mailfrom
 - servers/smtp/relay.c, 740
 - servers/smtp/smtp.h, 993
- smtp_client_send_nullfrom
 - servers/smtp/relay.c, 741
 - servers/smtp/smtp.h, 993
- smtp_client_send_rcptto
 - servers/smtp/relay.c, 741
 - servers/smtp/smtp.h, 993
- smtp_commands
 - servers/smtp/commands.h, 1636
- smtp_compare
 - servers/smtp/commands.c, 1620
 - servers/smtp/smtp.h, 994
- smtp_data
 - servers/smtp/smtp.h, 994
 - smtp.c, 2151
- smtp_data_finish
 - smtp.c, 2151
- smtp_data_inbound
 - smtp.c, 2151
- smtp_data_outbound
 - smtp.c, 2151
- smtp_data_read
 - smtp.c, 2152
- smtp_disabled
 - servers/smtp/smtp.h, 994
 - smtp.c, 2152
- smtp_ehlo
 - servers/smtp/smtp.h, 994
 - smtp.c, 2152
- smtp_fetch_authorization
 - servers/smtp/datatier.c, 709
 - servers/smtp/smtp.h, 995
- smtp_fetch_autoreply
 - servers/smtp/datatier.c, 710
 - servers/smtp/smtp.h, 995
- smtp_fetch_inbound
 - servers/smtp/datatier.c, 710
 - servers/smtp/smtp.h, 996
- smtp_fetch_rollmessages
 - servers/smtp/datatier.c, 712
 - servers/smtp/smtp.h, 997
- smtp_forward_message
 - servers/smtp/smtp.h, 997
 - transmit.c, 2156
- smtp_free_inbound
 - servers/smtp/smtp.h, 997
 - smtp/session.c, 2102
- smtp_free_outbound
 - servers/smtp/smtp.h, 998

- smtp/session.c, 2103
- smtp_free_recipients
 - servers/smtp/smtp.h, 998
 - smtp/session.c, 2103
- smtp_get_action
 - servers/smtp/datatier.c, 712
 - servers/smtp/smtp.h, 998
- smtp_helo
 - servers/smtp/smtp.h, 999
 - smtp.c, 2152
- smtp_inbound_filter_t, 139
 - action, 139
 - expression, 139
 - field, 139
 - foldernum, 139
 - label, 139
 - location, 139
 - ruenum, 140
 - type, 140
- smtp_inbound_prefs_t, 141
 - address, 142
 - autoreply, 142
 - bounces, 142
 - daily_rcv_limit, 142
 - daily_rcv_limit_ip, 142
 - dkim, 142
 - dkimaction, 142
 - domain, 142
 - filters, 142
 - foldernum, 142
 - forwarded, 143
 - greylist, 143
 - greytime, 143
 - inbox, 143
 - local_size, 143
 - mark, 143
 - messagenum, 143
 - next, 143
 - outcome, 143
 - overquota, 144
 - phish, 144
 - phishaction, 144
 - quota, 144
 - rbl, 144
 - rblaction, 144
 - rcptto, 144
 - rcv_size_limit, 144
 - rollout, 144
 - secure, 145
 - signet, 145
 - signum, 145
 - spam, 145
 - spam_checked, 145
 - spamaction, 145
 - spamkey, 145
 - spamsig, 145
 - spf, 145
 - spfaction, 146
 - stor_size, 146
 - usernum, 146
 - virus, 146
 - virusaction, 146
- smtp_init
 - servers/smtp/smtp.h, 999
 - smtp.c, 2153
- smtp_insert_spamsig
 - servers/smtp/datatier.c, 712
 - servers/smtp/smtp.h, 999
- smtp_invalid
 - servers/smtp/smtp.h, 1000
 - smtp.c, 2153
- smtp_list_free_filter
 - servers/smtp/smtp.h, 1000
 - smtp/session.c, 2103
- smtp_mail_from
 - servers/smtp/smtp.h, 1000
 - smtp.c, 2153
- smtp_message_t, 147
 - date, 147
 - from, 147
 - header_length, 147
 - id, 147
 - subject, 147
 - text, 147
 - to, 148
- smtp_noop
 - servers/smtp/smtp.h, 1001
 - smtp.c, 2154
- smtp_outbound_prefs_t, 149
 - daily_send_limit, 149
 - domain, 149
 - importance, 149
 - recipients, 149
 - send_size_limit, 149
 - sent_today, 149
 - tls, 150
 - usernum, 150
- smtp_parse_auth
 - servers/smtp/parse.c, 2096
 - servers/smtp/smtp.h, 1001
- smtp_parse_helo_domain
 - servers/smtp/parse.c, 2096
 - servers/smtp/smtp.h, 1001
- smtp_parse_mail_from_path
 - servers/smtp/parse.c, 2097
 - servers/smtp/smtp.h, 1002
- smtp_parse_rcpt_to
 - servers/smtp/parse.c, 2097
 - servers/smtp/smtp.h, 1002
- smtp_process
 - servers/smtp/commands.c, 1620
 - servers/smtp/smtp.h, 1002
- smtp_quit
 - servers/smtp/smtp.h, 1003

- smtp.c, 2154
- smtp_rcpt_to
 - servers/smtp/smtp.h, 1003
 - smtp.c, 2154
- smtp_recipients_t, 151
 - address, 151
 - next, 151
- smtp_relay_message
 - servers/smtp/smtp.h, 1003
 - transmit.c, 2156
- smtp_reply
 - servers/smtp/smtp.h, 1004
 - transmit.c, 2157
- smtp_requeue
 - servers/smtp/commands.c, 1620
 - servers/smtp/smtp.h, 1004
- smtp_rollout
 - accept.c, 2145
 - servers/smtp/smtp.h, 1004
- smtp_rset
 - servers/smtp/smtp.h, 1004
 - smtp.c, 2155
- SMTP_SELECT_USER_AUTH
 - queries.h, 2078
- smtp_send_message
 - servers/smtp/smtp.h, 1005
 - transmit.c, 2157
- smtp_session_destroy
 - servers/smtp/smtp.h, 1005
 - smtp/session.c, 2103
- smtp_session_reset
 - servers/smtp/smtp.h, 1005
 - smtp/session.c, 2104
- smtp_session_t, 152
 - authenticated, 152
 - bypass, 152
 - checked, 152
 - dkim, 152
 - esmtpl, 152
 - helo, 152
 - in_prefs, 153
 - ip_address_v4, 153
 - mailfrom, 153
 - max_length, 153
 - message, 153
 - num_recipients, 153
 - out_prefs, 153
 - rbl, 153
 - spf, 153
 - submission, 153
 - suggested_eight_bit, 153
 - suggested_length, 153
 - virus, 154
- smtp_sort
 - servers/smtp/commands.c, 1620
 - servers/smtp/smtp.h, 1006
- smtp_starttls
 - servers/smtp/smtp.h, 1006
 - smtp.c, 2155
- smtp_store_message
 - accept.c, 2146
 - servers/smtp/smtp.h, 1006
- smtp_store_spamsig
 - accept.c, 2146
 - servers/smtp/smtp.h, 1007
- smtp_update_receive_stats
 - servers/smtp/datatier.c, 713
 - servers/smtp/smtp.h, 1007
- smtp_update_transmission_stats
 - servers/smtp/datatier.c, 713
 - servers/smtp/smtp.h, 1007
- sockd
 - server_t, 131
- socket_path
 - magma_t, 100
- socket_timeout
 - server_config_t, 128
- sok_permissions_t
 - signet/common.h, 1578
- source
 - http_data_t, 76
- spam
 - server_config_t, 128
 - smtp_inbound_prefs_t, 145
- spam_checked
 - smtp_inbound_prefs_t, 145
- spamaction
 - smtp_inbound_prefs_t, 145
- spamkey
 - smtp_inbound_prefs_t, 145
- spamsig
 - smtp_inbound_prefs_t, 145
- special.h
 - bracket_extract_pl, 472
- spf
 - magma_t, 100
 - smtp_inbound_prefs_t, 145
 - smtp_session_t, 153
- spf.c
 - lib_load_spf, 1283
 - lib_version_spf, 1283
 - spf_check, 1283
 - spf_pool, 1284
 - spf_start, 1284
 - spf_stop, 1284
 - spf_version, 1284
- spf_check
 - checkers.h, 1267
 - spf.c, 1283
- SPF_dns_zone_add_str_d
 - symbols.h, 2054
- SPF_dns_zone_new_d
 - symbols.h, 2054
- SPF_get_lib_version_d

- symbols.h, 2054
- spf_pool
 - spf.c, 1284
- SPF_request_free_d
 - symbols.h, 2054
- SPF_request_new_d
 - symbols.h, 2055
- SPF_request_query_mailfrom_d
 - symbols.h, 2055
- SPF_request_set_env_from_d
 - symbols.h, 2055
- SPF_request_set_helo_dom_d
 - symbols.h, 2055
- SPF_request_set_ipv4_d
 - symbols.h, 2055
- SPF_request_set_ipv6_d
 - symbols.h, 2055
- SPF_response_free_d
 - symbols.h, 2055
- SPF_response_reason_d
 - symbols.h, 2055
- SPF_response_result_d
 - symbols.h, 2055
- SPF_server_free_d
 - symbols.h, 2055
- SPF_server_new_d
 - symbols.h, 2055
- spf_start
 - checkers.h, 1267
 - spf.c, 1284
- spf_stop
 - checkers.h, 1268
 - spf.c, 1284
- SPF_strerror_d
 - symbols.h, 2055
- SPF_strerror_d
 - symbols.h, 2056
- SPF_stresult_d
 - symbols.h, 2056
- spf_version
 - spf.c, 1284
- spfaction
 - smtp_inbound_prefs_t, 146
- spool
 - magma_t, 100
- spool.c
 - spool_check, 319
 - spool_check_file, 319
 - spool_cleanup, 320
 - spool_error_stats, 320
 - spool_mktemp, 320
 - spool_path, 321
 - spool_start, 321
 - spool_stop, 321
- spool_check
 - host.h, 301
 - spool.c, 319
- spool_check_file
 - host.h, 302
 - spool.c, 319
- spool_cleanup
 - host.h, 302
 - spool.c, 320
- spool_error_stats
 - host.h, 302
 - spool.c, 320
- spool_mktemp
 - host.h, 303
 - spool.c, 320
- spool_path
 - host.h, 303
 - spool.c, 321
- spool_start
 - host.h, 303
 - spool.c, 321
- spool_stop
 - host.h, 304
 - spool.c, 321
- sql
 - mysql.c, 1471
- sql_errno
 - database.h, 1454
 - mysql.c, 1468
- sql_error
 - database.h, 1455
 - mysql.c, 1468
- sql_insert
 - database.h, 1455
 - query.c, 1472
- sql_insert_conn
 - database.h, 1455
 - query.c, 1472
- sql_num_rows
 - database.h, 1455
 - query.c, 1472
- sql_num_rows_conn
 - database.h, 1455
 - query.c, 1472
- sql_open
 - database.h, 1455
 - mysql.c, 1468
- sql_ping
 - database.h, 1456
 - mysql.c, 1469
- sql_pool
 - database.h, 1465
 - dspam.c, 1281
 - mysql.c, 1471
- sql_query
 - database.h, 1456
 - query.c, 1472
- sql_query_conn
 - database.h, 1457
 - query.c, 1473

sql_query_res
 database.h, 1457
 query.c, 1473
sql_query_res_conn
 database.h, 1457
 query.c, 1473
sql_start
 database.h, 1457
 mysql.c, 1469
sql_stop
 database.h, 1458
 mysql.c, 1470
sql_thread_start
 database.h, 1458
 mysql.c, 1470
sql_thread_stop
 database.h, 1458
 mysql.c, 1470
sql_write
 database.h, 1458
 query.c, 1473
sql_write_conn
 database.h, 1458
 query.c, 1474
src/core/buckets/arrays.c, 161
src/core/buckets/buckets.h, 166
src/core/buckets/pool.c, 181
src/core/buckets/stacked.c, 187
src/core/checksum/adler.c, 189
src/core/checksum/checksum.h, 190
src/core/checksum/crc.c, 194
src/core/checksum/fletcher.c, 199
src/core/checksum/murmur.c, 200
src/core/classify/ascii.c, 201
src/core/classify/classify.h, 205
src/core/compare/compare.h, 209
src/core/compare/ends.c, 214
src/core/compare/equal.c, 215
src/core/compare/search.c, 217
src/core/compare/starts.c, 222
src/core/core.h, 223
src/core/encodings/base64.c, 232
src/core/encodings/encodings.h, 237
src/core/encodings/hex.c, 249
src/core/encodings/mappings.c, 253
src/core/encodings/qp.c, 254
src/core/encodings/url.c, 255
src/core/encodings/zbase32.c, 257
src/core/host/backtrace.c, 258
src/core/host/color.c, 259
src/core/host/errors.c, 265
src/core/host/files.c, 274
src/core/host/folder.c, 277
src/core/host/host.c, 278
src/core/host/host.h, 279
src/core/host/ip.c, 307
src/core/host/process.c, 314
src/core/host/signals.c, 318
src/core/host/spool.c, 319
src/core/host/tcp.c, 323
src/core/indexes/cursors.c, 326
src/core/indexes/hashed.c, 330
src/core/indexes/indexes.h, 336
src/core/indexes/inx.c, 348
src/core/indexes/linked.c, 355
src/core/memory/align.c, 364
src/core/memory/bitwise.c, 365
src/core/memory/memory.c, 370
src/core/memory/memory.h, 376
src/core/memory/secure.c, 386
src/core/parsers/bitwise.c, 367
src/core/parsers/case.c, 392
src/core/parsers/formats/formats.h, 394
src/core/parsers/formats/nvp.c, 396
src/core/parsers/line.c, 398
src/core/parsers/numbers/clamp.c, 400
src/core/parsers/numbers/digits.c, 404
src/core/parsers/numbers/numbers.c, 407
src/core/parsers/numbers/numbers.h, 422
src/core/parsers/parsers.h, 442
src/core/parsers/special/bracket.c, 471
src/core/parsers/special/special.h, 472
src/core/parsers/time.c, 473
src/core/parsers/token.c, 475
src/core/parsers/trim.c, 482
src/core/strings/allocation.c, 484
src/core/strings/data.c, 493
src/core/strings/info.c, 503
src/core/strings/length.c, 506
src/core/strings/multi.c, 510
src/core/strings/nuller.c, 516
src/core/strings/opts.c, 521
src/core/strings/print.c, 523
src/core/strings/replace.c, 527
src/core/strings/shortcuts.c, 528
src/core/strings/strings.h, 533
src/core/strings/validate.c, 579
src/core/thread/keys.c, 583
src/core/thread/mutex.c, 587
src/core/thread/rwlock.c, 590
src/core/thread/thread.c, 594
src/core/thread/thread.h, 599
src/core/type.c, 610
src/engine/config/cache/cache.c, 611
src/engine/config/cache/cache.h, 644
src/engine/config/cache/keys.h, 663
src/engine/config/config.h, 672
src/engine/config/global/datatier.c, 682
src/engine/config/global/global.c, 722
src/engine/config/global/global.h, 729
src/engine/config/global/keys.h, 664
src/engine/config/relay/keys.h, 665
src/engine/config/relay/relay.c, 735
src/engine/config/relay/relay.h, 742

src/engine/config/servers/keys.h, 666
 src/engine/config/servers/servers.c, 746
 src/engine/config/servers/servers.h, 752
 src/engine/context/args.c, 760
 src/engine/context/context.h, 762
 src/engine/context/process.c, 316
 src/engine/context/sanity.c, 770
 src/engine/context/signal.c, 771
 src/engine/context/system.c, 774
 src/engine/context/thread.c, 598
 src/engine/controller/controller.h, 777
 src/engine/controller/protocol.c, 781
 src/engine/controller/queue.c, 783
 src/engine/engine.h, 787
 src/engine/log/log.c, 788
 src/engine/log/log.h, 791
 src/engine/status/build.c, 795
 src/engine/status/performance.c, 797
 src/engine/status/statistics.c, 798
 src/engine/status/status.c, 807
 src/engine/status/status.h, 810
 src/magma.c, 821
 src/magma.h, 822
 src/network/addresses.c, 825
 src/network/clients.c, 829
 src/network/connections.c, 831
 src/network/dmtp.h, 835
 src/network/http.h, 850
 src/network/imap.h, 872
 src/network/listeners.c, 903
 src/network/meta.h, 905
 src/network/network.h, 931
 src/network/options.c, 952
 src/network/pop.h, 955
 src/network/read.c, 967
 src/network/reverse.c, 969
 src/network/sessions.h, 971
 src/network/smtp.h, 981
 src/network/write.c, 1009
 src/objects/auth/auth.c, 1013
 src/objects/auth/auth.h, 1016
 src/objects/auth/datatier.c, 683
 src/objects/auth/legacy.c, 1024
 src/objects/auth/stacie.c, 1025
 src/objects/auth/username.c, 1028
 src/objects/config/config.c, 1029
 src/objects/config/config.h, 677
 src/objects/config/datatier.c, 685
 src/objects/contacts/contacts.c, 1034
 src/objects/contacts/contacts.h, 1044
 src/objects/contacts/datatier.c, 687
 src/objects/contacts/find.c, 1055
 src/objects/folders/contacts.c, 1040
 src/objects/folders/datatier.c, 691
 src/objects/folders/find.c, 1057
 src/objects/folders/folders.c, 1059
 src/objects/folders/folders.h, 1073
 src/objects/folders/messages.c, 1084
 src/objects/locks.c, 1094
 src/objects/mail/cache.c, 615
 src/objects/mail/cleanup.c, 1097
 src/objects/mail/counters.c, 1099
 src/objects/mail/datatier.c, 693
 src/objects/mail/headers.c, 1101
 src/objects/mail/load_message.c, 1109
 src/objects/mail/mail.h, 1111
 src/objects/mail/mime.c, 1145
 src/objects/mail/objects.c, 1155
 src/objects/mail/parsing.c, 1161
 src/objects/mail/paths.c, 1162
 src/objects/mail/remove_message.c, 1164
 src/objects/mail/signatures.c, 1165
 src/objects/mail/store_message.c, 1172
 src/objects/messages/datatier.c, 696
 src/objects/messages/messages.c, 1086
 src/objects/messages/messages.h, 1175
 src/objects/messages/meta.c, 1188
 src/objects/meta/alerts.c, 1194
 src/objects/meta/alias.c, 1195
 src/objects/meta/crypto.c, 1196
 src/objects/meta/datatier.c, 698
 src/objects/meta/folders.c, 1062
 src/objects/meta/indexes.c, 1209
 src/objects/meta/locking.c, 1210
 src/objects/meta/meta.c, 1192
 src/objects/meta/meta.h, 908
 src/objects/meta/references.c, 1212
 src/objects/meta/serials.c, 1215
 src/objects/meta/updaters.c, 1220
 src/objects/objects.c, 1157
 src/objects/objects.h, 1224
 src/objects/serials.c, 1217
 src/objects/sessions/sessions.c, 1229
 src/objects/sessions/sessions.h, 973
 src/objects/warehouse/datatier.c, 707
 src/objects/warehouse/domains.c, 1244
 src/objects/warehouse/patterns.c, 1248
 src/objects/warehouse/warehouse.c, 1250
 src/objects/warehouse/warehouse.h, 1252
 src/providers/checkers/allocations.h, 1259
 src/providers/checkers/checkers.h, 1260
 src/providers/checkers/clamav.c, 1271
 src/providers/checkers/dkim.c, 1276
 src/providers/checkers/dspam.c, 1280
 src/providers/checkers/spf.c, 1283
 src/providers/compress/bzip.c, 1286
 src/providers/compress/compress.c, 1288
 src/providers/compress/compress.h, 1292
 src/providers/compress/engine.c, 1302
 src/providers/compress/lzo.c, 1303
 src/providers/compress/zlib.c, 1305
 src/providers/consumers/cache.c, 618
 src/providers/consumers/consumers.h, 1307
 src/providers/consumers/counters.c, 1100

src/providers/consumers/deserialization.c, 1321
src/providers/consumers/serialization.c, 1326
src/providers/cryptography/ciphers.c, 1331
src/providers/cryptography/cryptex.c, 1334
src/providers/cryptography/cryptography.h, 1342
src/providers/cryptography/digest.c, 1389
src/providers/cryptography/ecies.c, 1390
src/providers/cryptography/hash.c, 1402
src/providers/cryptography/hmac.c, 1404
src/providers/cryptography/openssl.c, 1408
src/providers/cryptography/parameters.c, 1414
src/providers/cryptography/random.c, 1417
src/providers/cryptography/scramble.c, 1422
src/providers/cryptography/symmetric.c, 1432
src/providers/cryptography/tls.c, 1436
src/providers/database/database.h, 1442
src/providers/database/mysql.c, 1466
src/providers/database/query.c, 1472
src/providers/database/results.c, 1475
src/providers/database/stmts.c, 1485
src/providers/database/transaction.c, 1492
src/providers/deprecated/cryptex.c, 1338
src/providers/deprecated/deprecated.h, 1494
src/providers/deprecated/ecies.c, 1396
src/providers/deprecated/hmac.c, 1406
src/providers/deprecated/scramble.c, 1427
src/providers/deprecated/symmetric.c, 1434
src/providers/dime/common/crypto.c, 1197
src/providers/dime/common/crypto_pub.c, 1510
src/providers/dime/common/dcrypto.h, 1514
src/providers/dime/common/error.c, 1519
src/providers/dime/common/error.h, 1524
src/providers/dime/common/misc.c, 1536
src/providers/dime/common/misc.h, 1550
src/providers/dime/common/misc_pub.c, 1556
src/providers/dime/common/network.c, 1561
src/providers/dime/common/network.h, 951
src/providers/dime/common/network_pub.c, 1563
src/providers/dime/dime_ctx.c, 1564
src/providers/dime/dime_ctx.h, 1566
src/providers/dime/dmessage/common.h, 1569
src/providers/dime/dmessage/crypto.h, 1580
src/providers/dime/dmessage/dmsg.c, 1590
src/providers/dime/dmessage/parse.h, 1599
src/providers/dime/dmessage/parser.c, 1603
src/providers/dime/dmtp/commands.c, 1606
src/providers/dime/dmtp/commands.h, 1622
src/providers/dime/ed25519/curve25519-donna-32bit.h, 1637
src/providers/dime/ed25519/curve25519-donna-64bit.h, 1639
src/providers/dime/ed25519/curve25519-donna-helpers.h, 1641
src/providers/dime/ed25519/curve25519-donna-sse2.h, 1642
src/providers/dime/ed25519/ed25519-donna-32bit-sse2.h, 1644
src/providers/dime/ed25519/ed25519-donna-32bit-tables.h, 1645
src/providers/dime/ed25519/ed25519-donna-64bit-sse2.h, 1646
src/providers/dime/ed25519/ed25519-donna-64bit-tables.h, 1647
src/providers/dime/ed25519/ed25519-donna-64bit-x86-32bit.h, 1648
src/providers/dime/ed25519/ed25519-donna-64bit-x86.h, 1649
src/providers/dime/ed25519/ed25519-donna-basepoint-table.h, 1650
src/providers/dime/ed25519/ed25519-donna-batchverify.h, 1651
src/providers/dime/ed25519/ed25519-donna-impl-base.h, 1653
src/providers/dime/ed25519/ed25519-donna-impl-sse2.h, 1654
src/providers/dime/ed25519/ed25519-donna-portable-identify.h, 1655
src/providers/dime/ed25519/ed25519-donna-portable.h, 1656
src/providers/dime/ed25519/ed25519-donna.h, 1657
src/providers/dime/ed25519/ed25519-hash-custom.h, 1661
src/providers/dime/ed25519/ed25519-hash.h, 1662
src/providers/dime/ed25519/ed25519-randombytes-custom.h, 1663
src/providers/dime/ed25519/ed25519-randombytes.h, 1664
src/providers/dime/ed25519/ed25519.c, 1665
src/providers/dime/ed25519/ed25519.h, 1669
src/providers/dime/ed25519/fuzz/curve25519-ref10.c, 1671
src/providers/dime/ed25519/fuzz/curve25519-ref10.h, 1674
src/providers/dime/ed25519/fuzz/ed25519-donna-sse2.c, 1675
src/providers/dime/ed25519/fuzz/ed25519-donna.c, 1676
src/providers/dime/ed25519/fuzz/ed25519-donna.h, 1658
src/providers/dime/ed25519/fuzz/ed25519-ref10.c, 1677
src/providers/dime/ed25519/fuzz/ed25519-ref10.h, 1679
src/providers/dime/ed25519/fuzz/fuzz-curve25519.c, 1680
src/providers/dime/ed25519/fuzz/fuzz-ed25519.c, 1681
src/providers/dime/ed25519/modm-donna-32bit.h, 1683
src/providers/dime/ed25519/modm-donna-64bit.h, 1684
src/providers/dime/ed25519/regression.h, 1685
src/providers/dime/ed25519/test-internals.c, 1686
src/providers/dime/ed25519/test-ticks.h, 1687
src/providers/dime/ed25519/test.c, 1688
src/providers/dime/error_codes.c, 1690
src/providers/dime/error_codes.h, 1691
src/providers/dime/sds/sds.c, 1693
src/providers/dime/sds/sds.h, 1699
src/providers/dime/sds/sdsalloc.h, 1706
src/providers/dime/sds/testhelp.h, 1707
src/providers/dime/signet-resolver/cache.c, 624
src/providers/dime/signet-resolver/cache.h, 648
src/providers/dime/signet-resolver/cache_pub.c, 1708
src/providers/dime/signet-resolver/dmtp.c, 1711
src/providers/dime/signet-resolver/dmtp.h, 836
src/providers/dime/signet-resolver/dmtp_pub.c, 1727
src/providers/dime/signet-resolver/dns.c, 1730
src/providers/dime/signet-resolver/dns.h, 1746
src/providers/dime/signet-resolver/mrec.c, 1760
src/providers/dime/signet-resolver/mrec.h, 1764
src/providers/dime/signet-resolver/mrec_pub.c, 1768
src/providers/dime/signet-resolver/signet-ssl.h, 1769
src/providers/dime/signet-resolver/ssl.c, 1775
src/providers/dime/signet-resolver/ssl_pub.c, 1784
src/providers/dime/signet/common.h, 1573
src/providers/dime/signet/general.c, 1786
src/providers/dime/signet/keys.c, 585
src/providers/dime/signet/keys.h, 667
src/providers/dime/signet/signet.c, 1788
src/providers/dime/signet/signet.h, 1806
src/providers/dime/util/encoding.c, 1825
src/providers/dime/util/encoding.h, 1826

src/providers/dime/util/encrypt.c, 1827
 src/providers/dime/util/encrypt.h, 1830
 src/providers/images/freetype.c, 1832
 src/providers/images/gd.c, 1833
 src/providers/images/images.h, 1834
 src/providers/images/jpeg.c, 1837
 src/providers/images/png.c, 1838
 src/providers/parsers/json.c, 1839
 src/providers/parsers/parsers.h, 456
 src/providers/parsers/utf8.c, 1840
 src/providers/parsers/xml.c, 1842
 src/providers/prime/cryptography/aes.c, 1855
 src/providers/prime/cryptography/cryptography.h, 1381
 src/providers/prime/cryptography/ed25519.c, 1666
 src/providers/prime/cryptography/secp256k1.c, 1859
 src/providers/prime/keys/keys.h, 669
 src/providers/prime/keys/orgs.c, 1863
 src/providers/prime/keys/users.c, 1867
 src/providers/prime/messages/chunks/chunks.c, 1871
 src/providers/prime/messages/chunks/chunks.h, 1873
 src/providers/prime/messages/chunks/encrypted.c, 1882
 src/providers/prime/messages/chunks/ephemeral.c, 1884
 src/providers/prime/messages/chunks/keys.c, 1886
 src/providers/prime/messages/chunks/signature.c, 1888
 src/providers/prime/messages/chunks/slots.c, 1890
 src/providers/prime/messages/messages.c, 1088
 src/providers/prime/messages/messages.h, 1186
 src/providers/prime/messages/parts/parts.c, 1893
 src/providers/prime/messages/parts/parts.h, 1894
 src/providers/prime/prime.c, 1895
 src/providers/prime/prime.h, 1900
 src/providers/prime/signets/orgs.c, 1865
 src/providers/prime/signets/requests.c, 1913
 src/providers/prime/signets/signets.h, 1915
 src/providers/prime/signets/users.c, 1869
 src/providers/prime/transposition/armored/armored.h, 1920
 src/providers/prime/transposition/armored/pem.c, 1921
 src/providers/prime/transposition/binary/binary.h, 1924
 src/providers/prime/transposition/binary/fields.c, 1932
 src/providers/prime/transposition/binary/headers.c, 1106
 src/providers/prime/transposition/binary/objects.c, 1159
 src/providers/prime/transposition/binary/reader.c, 1934
 src/providers/prime/transposition/binary/unpack.c, 1936
 src/providers/prime/transposition/transposition.h, 1937
 src/providers/providers.h, 1938
 src/providers/stacie/creation.c, 1940
 src/providers/stacie/crypto.c, 1207
 src/providers/stacie/passwords.c, 1942
 src/providers/stacie/realms.c, 1945
 src/providers/stacie/stacie.h, 1947
 src/providers/stacie/tokens.c, 1956
 src/providers/storage/data.c, 499
 src/providers/storage/storage.h, 1958
 src/providers/storage/tank.c, 1966
 src/providers/storage/tokyo.c, 1971
 src/providers/storage/tree.c, 1972
 src/providers/symbols.c, 1977
 src/providers/symbols.h, 2010
 src/queries.h, 2068
 src/servers/dmtp/commands.c, 1614
 src/servers/dmtp/commands.h, 1631
 src/servers/dmtp/dmtp.c, 1722
 src/servers/dmtp/dmtp.h, 844
 src/servers/dmtp/parse.c, 2083
 src/servers/dmtp/session.c, 2100
 src/servers/http/commands.h, 1632
 src/servers/http/content.c, 2105
 src/servers/http/data.c, 500
 src/servers/http/errors.c, 270
 src/servers/http/http.c, 2110
 src/servers/http/http.h, 852
 src/servers/http/parse.c, 2084
 src/servers/http/response.c, 2112
 src/servers/http/sessions.c, 1238
 src/servers/imap/commands.c, 1616
 src/servers/imap/commands.h, 1633
 src/servers/imap/fetch.c, 2115
 src/servers/imap/fetch_response.c, 2120
 src/servers/imap/flags.c, 2121
 src/servers/imap/folders.c, 1065
 src/servers/imap/imap.c, 2123
 src/servers/imap/imap.h, 873
 src/servers/imap/messages.c, 1090
 src/servers/imap/output.c, 2130
 src/servers/imap/parse.c, 2088
 src/servers/imap/parse_address.c, 2131
 src/servers/imap/range.c, 2132
 src/servers/imap/search.c, 219
 src/servers/imap/sessions.c, 1239
 src/servers/molten/commands.c, 1618
 src/servers/molten/commands.h, 1634
 src/servers/molten/molten.c, 2133
 src/servers/molten/molten.h, 2134
 src/servers/molten/sessions.c, 1240
 src/servers/pop/commands.c, 1619
 src/servers/pop/commands.h, 1635
 src/servers/pop/mailbox.c, 2136
 src/servers/pop/parse.c, 2094
 src/servers/pop/pop.c, 2138
 src/servers/pop/pop.h, 956
 src/servers/pop/sessions.c, 1241
 src/servers/servers.h, 759
 src/servers/smtp/accept.c, 2145
 src/servers/smtp/checkers.c, 2147
 src/servers/smtp/commands.c, 1620
 src/servers/smtp/commands.h, 1636
 src/servers/smtp/datatier.c, 708
 src/servers/smtp/parse.c, 2096
 src/servers/smtp/relay.c, 739
 src/servers/smtp/session.c, 2101
 src/servers/smtp/smtp.c, 2150
 src/servers/smtp/smtp.h, 983
 src/servers/smtp/transmit.c, 2156
 src/web/contact/abuse.c, 2158

src/web/contact/business.c, 2162
 src/web/contact/contact.c, 2166
 src/web/contact/contact.h, 2168
 src/web/json_api/endpoints.c, 2172
 src/web/json_api/helpers.c, 2173
 src/web/json_api/json_api.c, 2175
 src/web/json_api/json_api.h, 2176
 src/web/portal/config.c, 1032
 src/web/portal/contacts.c, 1043
 src/web/portal/endpoint.c, 2179
 src/web/portal/flags.c, 2122
 src/web/portal/folders.c, 1070
 src/web/portal/mail.c, 2199
 src/web/portal/messages.c, 1091
 src/web/portal/methods.h, 2202
 src/web/portal/parse.c, 2098
 src/web/portal/portal.c, 2203
 src/web/portal/portal.h, 2205
 src/web/register/abuse.c, 2159
 src/web/register/business.c, 2164
 src/web/register/captcha.c, 2238
 src/web/register/datatier.c, 715
 src/web/register/register.c, 2240
 src/web/register/register.h, 2243
 src/web/register/sessions.c, 1242
 src/web/statistics/datatier.c, 717
 src/web/statistics/statistics.c, 806
 src/web/statistics/statistics.h, 2254
 src/web/teacher/datatier.c, 719
 src/web/teacher/teacher.c, 2256
 src/web/teacher/teacher.h, 2259
 src/web/web.h, 2264
 SS_BROKEN_COC
 signet.h, 1809
 SS_CRYPT0
 signet.h, 1809
 SS_FULL
 signet.h, 1809
 SS_ID
 signet.h, 1809
 SS_INCOMPLETE
 signet.h, 1809
 SS_INVALID
 signet.h, 1809
 SS_MALFORMED
 signet.h, 1809
 SS_OVERFLOW
 signet.h, 1809
 SS_SSR
 signet.h, 1809
 SS_UNKNOWN
 signet.h, 1809
 ssize_conv_bl
 numbers.c, 414
 numbers.h, 432
 ssl.c
 _deserialize_ocsp_response_cb, 1776
 _destroy_ocsp_response_cb, 1776
 _do_ocsp_validation, 1777
 _do_x509_hostname_check, 1777
 _do_x509_validation, 1778
 _domain_wildcard_check, 1778
 _dump_ocsp_response_cb, 1778
 _get_cache_ocsp_id, 1779
 _get_cert_store, 1779
 _get_cert_subject_cn, 1779
 _ocsp_response_callback, 1780
 _serialize_ocsp_response_cb, 1780
 _ssl_connect_host, 1780
 _ssl_disconnect, 1781
 _ssl_fd_loop, 1781
 _ssl_get_client_context, 1781
 _ssl_initialize, 1782
 _ssl_shutdown, 1782
 _ssl_starttls, 1782
 _validate_self_signed, 1782
 _verify_certificate_callback, 1783
 SSL_accept_d
 symbols.c, 2005
 symbols.h, 2056
 SSL_CIPHER_get_bits_d
 symbols.h, 2056
 SSL_CIPHER_get_name_d
 symbols.c, 2005
 symbols.h, 2056
 SSL_CIPHER_get_version_d
 symbols.c, 2005
 symbols.h, 2056
 SSL_connect_d
 symbols.c, 2006
 symbols.h, 2056
 ssl_connect_host
 ssl_pub.c, 1784
 SSL_ctrl_d
 symbols.c, 2006
 symbols.h, 2056
 SSL_CTX_callback_ctrl_d
 symbols.c, 2006
 symbols.h, 2056
 SSL_CTX_check_private_key_d
 symbols.c, 2006
 symbols.h, 2056
 SSL_CTX_ctrl_d
 symbols.h, 2056
 SSL_CTX_free_d
 symbols.c, 2006
 symbols.h, 2056
 SSL_CTX_load_verify_locations_d
 symbols.h, 2056
 SSL_CTX_new_d
 symbols.c, 2006
 symbols.h, 2057
 SSL_CTX_set_cipher_list_d
 symbols.c, 2006

- symbols.h, 2057
- SSL_CTX_set_tmp_dh_callback_d
 - symbols.h, 2057
- SSL_CTX_set_tmp_ecdh_callback_d
 - symbols.h, 2057
- SSL_CTX_set_verify_d
 - symbols.h, 2057
- SSL_CTX_use_certificate_chain_file_d
 - symbols.h, 2057
- SSL_CTX_use_PrivateKey_file_d
 - symbols.h, 2057
- ssl_dh2048_exchange_callback
 - cryptography/cryptography.h, 1372
- ssl_dh4096_exchange_callback
 - cryptography/cryptography.h, 1372
- ssl_dh_generate_callback
 - cryptography/cryptography.h, 1372
- ssl_disconnect
 - ssl_pub.c, 1784
- SSL_do_handshake_d
 - symbols.c, 2006
 - symbols.h, 2057
- ssl_ecdh_exchange_callback
 - cryptography/cryptography.h, 1372
 - openssl.c, 1409
- ssl_error_string
 - cryptography/cryptography.h, 1372
 - openssl.c, 1409
- SSL_free_d
 - symbols.c, 2006
 - symbols.h, 2057
- ssl_get_client_context
 - ssl_pub.c, 1784
- SSL_get_current_cipher_d
 - symbols.c, 2006
 - symbols.h, 2057
- SSL_get_error_d
 - symbols.c, 2006
 - symbols.h, 2057
- SSL_get_fd_d
 - symbols.c, 2006
 - symbols.h, 2057
- SSL_get_peer_cert_chain_d
 - symbols.h, 2057
- SSL_get_peer_certificate_d
 - symbols.c, 2006
 - symbols.h, 2058
- SSL_get_read_ahead_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_get_rfd_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_get_shutdown_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_get_version_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_get_wbio_d
 - symbols.c, 2007
 - symbols.h, 2058
- ssl_initialize
 - ssl_pub.c, 1785
- SSL_library_init_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_load_error_strings_d
 - symbols.c, 2007
 - symbols.h, 2058
- ssl_locking_callback
 - cryptography/cryptography.h, 1373
 - openssl.c, 1410
- ssl_locks
 - openssl.c, 1412
- SSL_new_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_peek_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_pending_d
 - symbols.c, 2007
 - symbols.h, 2058
- ssl_pub.c
 - do_ocsp_validation, 1784
 - do_x509_hostname_check, 1784
 - do_x509_validation, 1784
 - get_cert_subject_cn, 1784
 - ssl_connect_host, 1784
 - ssl_disconnect, 1784
 - ssl_get_client_context, 1784
 - ssl_initialize, 1785
 - ssl_shutdown, 1785
 - ssl_starttls, 1785
- SSL_read_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_set_accept_state_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_set_bio_d
 - symbols.c, 2007
 - symbols.h, 2058
- SSL_set_connect_state_d
 - symbols.c, 2007
 - symbols.h, 2059
- SSL_set_fd_d
 - symbols.c, 2008
 - symbols.h, 2059
- SSL_set_read_ahead_d
 - symbols.c, 2008
 - symbols.h, 2059
- ssl_shutdown

- ssl_pub.c, 1785
- SSL_shutdown_d
 - symbols.c, 2008
 - symbols.h, 2059
- ssl_start
 - cryptography/cryptography.h, 1373
 - openssl.c, 1410
- ssl_starttls
 - ssl_pub.c, 1785
- ssl_stop
 - cryptography/cryptography.h, 1373
 - openssl.c, 1411
- ssl_thread_id_callback
 - cryptography/cryptography.h, 1374
 - openssl.c, 1411
- ssl_thread_stop
 - cryptography/cryptography.h, 1374
 - openssl.c, 1411
- ssl_verify_privkey
 - cryptography/cryptography.h, 1374
 - openssl.c, 1411
- ssl_version
 - openssl.c, 1412
- SSL_version_str_d
 - symbols.c, 2008
 - symbols.h, 2059
- SSL_want_d
 - symbols.c, 2008
 - symbols.h, 2059
- SSL_write_d
 - symbols.c, 2008
 - symbols.h, 2059
- SSLeay_version_d
 - symbols.c, 2008
 - symbols.h, 2059
- SSLv23_client_method_d
 - symbols.c, 2008
 - symbols.h, 2059
- SSLv23_server_method_d
 - symbols.c, 2008
 - symbols.h, 2059
- st
 - multi_t, 112
- st_alloc
 - allocation.c, 485
 - strings.h, 555
- st_alloc_opts
 - allocation.c, 485
 - strings.h, 556
- st_and
 - core/parsers/parsers.h, 448
 - parsers/bitwise.c, 367
- st_append
 - strings.h, 541
- st_append_opts
 - allocation.c, 486
 - strings.h, 556
- st_append_out
 - allocation.c, 486
 - strings.h, 557
- st_aprint
 - print.c, 523
 - strings.h, 557
- st_aprint_opts
 - print.c, 523
 - strings.h, 557
- st_avail_get
 - length.c, 506
 - strings.h, 557
- st_avail_set
 - length.c, 506
 - strings.h, 557
- st_bitwise
 - parsers/bitwise.c, 367
- st_char_get
 - core/strings/data.c, 493
 - strings.h, 558
- st_cleanup
 - strings.h, 541
- st_cleanup_variadic
 - allocation.c, 486
 - strings.h, 559
- st_cmp_ci_ends
 - compare.h, 210
 - ends.c, 214
- st_cmp_ci_eq
 - compare.h, 210
 - equal.c, 216
- st_cmp_ci_starts
 - compare.h, 211
 - starts.c, 222
- st_cmp_cs_ends
 - compare.h, 211
 - ends.c, 214
- st_cmp_cs_eq
 - compare.h, 212
 - equal.c, 216
- st_cmp_cs_starts
 - compare.h, 212
 - starts.c, 222
- st_copy_in
 - allocation.c, 487
 - strings.h, 559
- st_data_get
 - core/strings/data.c, 495
 - strings.h, 560
- st_data_set
 - core/strings/data.c, 495
 - strings.h, 560
- st_dupe
 - allocation.c, 487
 - strings.h, 561
- st_dupe_opts
 - allocation.c, 487

- strings.h, 561
- st_empty
 - strings.h, 542
- st_empty_out
 - core/strings/data.c, 496
 - strings.h, 562
- st_empty_variadic
 - core/strings/data.c, 496
 - strings.h, 562
- st_free
 - allocation.c, 488
 - strings.h, 563
- st_import
 - allocation.c, 489
 - strings.h, 564
- st_import_opts
 - allocation.c, 490
 - strings.h, 564
- st_info_allocator
 - info.c, 503
 - strings.h, 565
- st_info_layout
 - info.c, 503
 - strings.h, 565
- st_info_opts
 - info.c, 504
 - strings.h, 565
- st_info_type
 - info.c, 504
 - strings.h, 566
- st_length_get
 - length.c, 507
 - strings.h, 566
- st_length_int
 - length.c, 508
 - strings.h, 567
- st_length_set
 - length.c, 509
 - strings.h, 568
- st_merge
 - strings.h, 542
- st_merge_opts
 - allocation.c, 490
 - strings.h, 569
- st_not
 - core/parsers/parsers.h, 448
 - parsers/bitwise.c, 368
- st_nullify
 - allocation.c, 490
 - strings.h, 569
- st_opt_get
 - opts.c, 521
 - strings.h, 569
- st_opt_set
 - opts.c, 521
 - strings.h, 570
- st_opt_test
 - opts.c, 521
 - strings.h, 570
- st_option_allocators
 - info.c, 504
- st_option_flags
 - info.c, 505
- st_option_layouts
 - info.c, 505
- st_option_types
 - info.c, 505
- st_or
 - core/parsers/parsers.h, 449
 - parsers/bitwise.c, 368
- st_output
 - allocation.c, 491
 - strings.h, 570
- st_populated
 - strings.h, 542
- st_populated_variadic
 - core/strings/data.c, 496
 - strings.h, 571
- st_quick
 - print.c, 524
 - strings.h, 571
- st_realloc
 - allocation.c, 491
 - strings.h, 571
- st_replace
 - replace.c, 527
 - strings.h, 572
- st_search_chr
 - compare.h, 212
 - core/compare/search.c, 217
- st_search_ci
 - compare.h, 213
 - core/compare/search.c, 217
- st_search_cs
 - compare.h, 213
 - core/compare/search.c, 217
- st_set
 - core/strings/data.c, 497
 - strings.h, 572
- st_sprint
 - print.c, 524
 - strings.h, 572
- st_swap
 - replace.c, 527
 - strings.h, 572
- st_trim
 - core/parsers/parsers.h, 449
 - trim.c, 482
- st_uchar_get
 - core/strings/data.c, 497
 - strings.h, 573
- st_used_variadic
 - strings.h, 573
- st_valid_append

- strings.h, 573
- validate.c, 579
- st_valid_avail
 - strings.h, 573
 - validate.c, 579
- st_valid_destination
 - strings.h, 574
 - validate.c, 580
- st_valid_free
 - strings.h, 574
 - validate.c, 580
- st_valid_joined
 - strings.h, 574
 - validate.c, 580
- st_valid_opts
 - strings.h, 575
 - validate.c, 581
- st_valid_placer
 - strings.h, 575
 - validate.c, 581
- st_valid_tracked
 - strings.h, 575
 - validate.c, 581
- st_vaprint
 - strings.h, 542
- st_vaprint_opts
 - print.c, 525
 - strings.h, 576
- st_vsprint
 - print.c, 525
 - strings.h, 576
- st_wipe
 - core/strings/data.c, 497
 - strings.h, 577
- st_write
 - strings.h, 542
- st_write_variadic
 - print.c, 525
 - strings.h, 577
- st_xor
 - core/parsers/parsers.h, 449
 - parsers/bitwise.c, 368
- stacie.c
 - auth_stacie, 1025
 - auth_stacie_alloc, 1025
 - auth_stacie_cleanup, 1026
 - auth_stacie_free, 1026
- stacie.h
 - STACIE_BLOCK_LENGTH, 1948
 - stacie_create_nonce, 1949
 - stacie_create_salt, 1950
 - stacie_create_shard, 1950
 - stacie_decrypt, 1950
 - stacie_derive_key, 1951
 - stacie_derive_rounds, 1952
 - stacie_derive_seed, 1952
 - stacie_derive_token, 1953
 - stacie_encrypt, 1953
 - STACIE_ENCRYPT_MAX, 1948
 - STACIE_ENCRYPT_MIN, 1948
 - STACIE_ENVELOPE_LENGTH, 1948
 - STACIE_KEY_LENGTH, 1948
 - STACIE_KEY_ROUNDS_MAX, 1948
 - STACIE_KEY_ROUNDS_MIN, 1949
 - STACIE_NONCE_LENGTH, 1949
 - stacie_realm_cipher, 1954
 - stacie_realm_key, 1954
 - stacie_realm_tag, 1955
 - stacie_realm_vector, 1955
 - STACIE_SALT_LENGTH, 1949
 - STACIE_SHARD_LENGTH, 1949
 - STACIE_TOKEN_LENGTH, 1949
 - STACIE_TOKEN_ROUNDS, 1949
 - STACIE_BLOCK_LENGTH
 - stacie.h, 1948
 - stacie_create_nonce
 - creation.c, 1940
 - stacie.h, 1949
 - stacie_create_salt
 - creation.c, 1940
 - stacie.h, 1950
 - stacie_create_shard
 - creation.c, 1941
 - stacie.h, 1950
 - stacie_decrypt
 - providers/stacie/crypto.c, 1207
 - stacie.h, 1950
 - stacie_derive_key
 - passwords.c, 1942
 - stacie.h, 1951
 - stacie_derive_rounds
 - passwords.c, 1943
 - stacie.h, 1952
 - stacie_derive_seed
 - passwords.c, 1943
 - stacie.h, 1952
 - stacie_derive_token
 - stacie.h, 1953
 - tokens.c, 1956
 - stacie_encrypt
 - providers/stacie/crypto.c, 1207
 - stacie.h, 1953
 - STACIE_ENCRYPT_MAX
 - stacie.h, 1948
 - STACIE_ENCRYPT_MIN
 - stacie.h, 1948
 - STACIE_ENVELOPE_LENGTH
 - stacie.h, 1948
 - STACIE_KEY_LENGTH
 - stacie.h, 1948
 - STACIE_KEY_ROUNDS_MAX
 - stacie.h, 1948
 - STACIE_KEY_ROUNDS_MIN
 - stacie.h, 1949

- STACIE_NONCE_LENGTH
 - stacie.h, 1949
- stacie_realm_cipher
 - realms.c, 1945
 - stacie.h, 1954
- stacie_realm_key
 - realms.c, 1945
 - stacie.h, 1954
- stacie_realm_tag
 - realms.c, 1946
 - stacie.h, 1955
- stacie_realm_vector
 - realms.c, 1946
 - stacie.h, 1955
- STACIE_SALT_LENGTH
 - stacie.h, 1949
- STACIE_SHARD_LENGTH
 - stacie.h, 1949
- STACIE_TOKEN_LENGTH
 - stacie.h, 1949
- STACIE_TOKEN_ROUNDS
 - stacie.h, 1949
- STACK
 - strings.h, 543
- stack
 - magma_t, 100
- STACK_OF
 - symbols.c, 1984
 - symbols.h, 2023
- stacked.c
 - stacker_alloc, 187
 - stacker_free, 187
 - stacker_nodes, 187
 - stacker_pop, 188
 - stacker_push, 188
- stacker_alloc
 - buckets.h, 178
 - stacked.c, 187
- stacker_free
 - buckets.h, 178
 - stacked.c, 187
- stacker_node_t
 - buckets.h, 179
- stacker_nodes
 - buckets.h, 178
 - stacked.c, 187
- stacker_pop
 - buckets.h, 179
 - stacked.c, 188
- stacker_push
 - buckets.h, 179
 - stacked.c, 188
- stacker_t
 - buckets.h, 180
- stamp_counter_check
 - providers/consumers/counters.c, 1100
- stamp_counter_increment
 - providers/consumers/counters.c, 1100
- standard
 - magma_t, 100
- starts.c
 - st_cmp_ci_starts, 222
 - st_cmp_cs_starts, 222
- startup
 - status.c, 809
- state
 - dmime_message_t, 47
 - dmime_object_t, 49
- statistics
 - magma_t, 100
- statistics.h
 - portal_stat_emails_received_today, 2254
 - portal_stat_emails_received_week, 2254
 - portal_stat_emails_sent_today, 2254
 - portal_stat_emails_sent_week, 2254
 - portal_stat_total_users, 2254
 - portal_stat_users_checked_email_today, 2254
 - portal_stat_users_checked_email_week, 2254
 - portal_stat_users_registered_today, 2254
 - portal_stat_users_registered_total, 2254
 - portal_stat_users_registered_week, 2254
 - portal_stat_users_sent_email_today, 2254
 - portal_stat_users_sent_email_week, 2254
 - statistics_init, 2255
 - statistics_process, 2255
 - statistics_refresh, 2255
- STATISTICS_GET_EMAILS_RECEIVED_TODAY
 - queries.h, 2078
- STATISTICS_GET_EMAILS_RECEIVED_WEEK
 - queries.h, 2078
- STATISTICS_GET_EMAILS_SENT_TODAY
 - queries.h, 2078
- STATISTICS_GET_EMAILS_SENT_WEEK
 - queries.h, 2078
- STATISTICS_GET_TOTAL_USERS
 - queries.h, 2079
- STATISTICS_GET_USERS_CHECKED_EMAIL_TODAY
 - queries.h, 2079
- STATISTICS_GET_USERS_CHECKED_EMAIL_WEEK
 - queries.h, 2079
- STATISTICS_GET_USERS_REGISTERED_TODAY
 - queries.h, 2079
- STATISTICS_GET_USERS_REGISTERED_WEEK
 - queries.h, 2079
- STATISTICS_GET_USERS_SENT_EMAIL_TODAY
 - queries.h, 2079
- STATISTICS_GET_USERS_SENT_EMAIL_WEEK
 - queries.h, 2079
- statistics_init
 - statistics.h, 2255
 - web/statistics/datatier.c, 717
- statistics_last_updated
 - web/statistics/datatier.c, 718
- statistics_process

- statistics.h, 2255
- web/statistics/statistics.c, 806
- statistics_refresh
 - statistics.h, 2255
 - web/statistics/datatier.c, 717
- statistics_vp_t, 155
- stmt, 155
- val, 155
- stats
 - engine/status/statistics.c, 805
- stats_adjust_by_name
 - engine/status/statistics.c, 799
 - status.h, 813
- stats_adjust_by_num
 - engine/status/statistics.c, 799
 - status.h, 813
- stats_decrement_by_name
 - engine/status/statistics.c, 800
 - status.h, 813
- stats_decrement_by_num
 - engine/status/statistics.c, 800
 - status.h, 814
- stats_derived_count
 - engine/status/statistics.c, 800
 - status.h, 814
- stats_derived_name
 - engine/status/statistics.c, 800
 - status.h, 814
- stats_derived_value
 - engine/status/statistics.c, 801
 - status.h, 814
- stats_get_count
 - engine/status/statistics.c, 801
 - status.h, 815
- stats_get_name
 - engine/status/statistics.c, 801
 - status.h, 815
- stats_get_name_pos
 - engine/status/statistics.c, 801
 - status.h, 815
- stats_get_value_by_name
 - engine/status/statistics.c, 802
 - status.h, 816
- stats_get_value_by_num
 - engine/status/statistics.c, 802
 - status.h, 816
- stats_increment_by_name
 - engine/status/statistics.c, 802
 - status.h, 816
- stats_increment_by_num
 - engine/status/statistics.c, 803
 - status.h, 816
- stats_init
 - engine/status/statistics.c, 803
 - status.h, 817
- stats_set_by_name
 - engine/status/statistics.c, 803
- status.h, 817
- stats_set_by_num
 - engine/status/statistics.c, 803
 - status.h, 817
- stats_shutdown
 - engine/status/statistics.c, 804
 - status.h, 818
- stats_sum_errors
 - engine/status/statistics.c, 804
 - status.h, 818
- status
 - allocations.h, 1259
 - auth_t, 26
 - status.c, 807
 - status.h, 818
- status.c
 - pid, 809
 - process, 809
 - startup, 809
 - status, 807
 - status_get, 807
 - status_level, 809
 - status_lock, 809
 - status_pid, 808
 - status_process, 808
 - status_set, 808
 - status_signal, 808
 - status_startup, 809
- status.h
 - build_commit, 812
 - build_stamp, 812
 - build_version, 812
 - build_version_major, 812
 - build_version_minor, 812
 - build_version_patch, 812
 - perf_rdtsc, 813
 - stats_adjust_by_name, 813
 - stats_adjust_by_num, 813
 - stats_decrement_by_name, 813
 - stats_decrement_by_num, 814
 - stats_derived_count, 814
 - stats_derived_name, 814
 - stats_derived_value, 814
 - stats_get_count, 815
 - stats_get_name, 815
 - stats_get_name_pos, 815
 - stats_get_value_by_name, 816
 - stats_get_value_by_num, 816
 - stats_increment_by_name, 816
 - stats_increment_by_num, 816
 - stats_init, 817
 - stats_set_by_name, 817
 - stats_set_by_num, 817
 - stats_shutdown, 818
 - stats_sum_errors, 818
 - status, 818
 - status_get, 818

- status_pid, 819
- status_process, 819
- status_set, 819
- status_signal, 819
- status_startup, 820
- status_get
 - status.c, 807
 - status.h, 818
- status_level
 - status.c, 809
- status_lock
 - status.c, 809
- status_pid
 - status.c, 808
 - status.h, 819
- status_process
 - status.c, 808
 - status.h, 819
- status_set
 - status.c, 808
 - status.h, 819
- status_signal
 - status.c, 808
 - status.h, 819
- status_startup
 - status.c, 809
 - status.h, 820
- status_t
 - buckets.h, 169
- stmt
 - statistics_vp_t, 155
- stmt_bind_param
 - database.h, 1459
 - stmts.c, 1486
- stmt_close
 - database.h, 1459
 - stmts.c, 1486
- stmt_errno
 - database.h, 1459
 - stmts.c, 1486
- stmt_error
 - database.h, 1459
 - stmts.c, 1486
- stmt_exec
 - database.h, 1459
 - stmts.c, 1486
- stmt_exec_affected
 - database.h, 1460
 - stmts.c, 1487
- stmt_exec_affected_conn
 - database.h, 1460
 - stmts.c, 1487
- stmt_exec_conn
 - database.h, 1461
 - stmts.c, 1488
- stmt_get_result
 - database.h, 1461
- stmts.c, 1488
- stmt_get_result_conn
 - database.h, 1462
 - stmts.c, 1489
- stmt_insert
 - database.h, 1462
 - stmts.c, 1489
- stmt_insert_conn
 - database.h, 1462
 - stmts.c, 1489
- stmt_open
 - database.h, 1463
 - stmts.c, 1490
- stmt_prepare
 - database.h, 1463
 - stmts.c, 1490
- stmt_rebuild
 - database.h, 1463
 - stmts.c, 1490
- stmt_reset
 - database.h, 1463
 - stmts.c, 1490
- stmt_start
 - database.h, 1464
 - stmts.c, 1490
- stmt_stop
 - database.h, 1464
 - stmts.c, 1491
- stmts.c
 - queries, 1491
 - stmt_bind_param, 1486
 - stmt_close, 1486
 - stmt_errno, 1486
 - stmt_error, 1486
 - stmt_exec, 1486
 - stmt_exec_affected, 1487
 - stmt_exec_affected_conn, 1487
 - stmt_exec_conn, 1488
 - stmt_get_result, 1488
 - stmt_get_result_conn, 1489
 - stmt_insert, 1489
 - stmt_insert_conn, 1489
 - stmt_open, 1490
 - stmt_prepare, 1490
 - stmt_rebuild, 1490
 - stmt_reset, 1490
 - stmt_start, 1490
 - stmt_stop, 1491
- STMTS_INIT
 - queries.h, 2079, 2082
- stor_size
 - smtp_inbound_prefs_t, 146
- storage
 - magma_t, 101
- storage.h
 - __attribute__, 1959
 - entry_t, 1964

- lib_load_tokyo, 1960
- lib_version_tokyo, 1960
- record_t, 1964
- TANK_COMPRESS_BZIP, 1959
- TANK_COMPRESS_LZO, 1959
- TANK_COMPRESS_ZLIB, 1959
- tank_count, 1960
- tank_cycle, 1960
- tank_delete, 1961
- tank_delete_object, 1961
- TANK_ENTRY_VERSION, 1959
- TANK_FLAGS_E, 1965
- tank_insert_object, 1961
- tank_load, 1962
- tank_maintain, 1962
- TANK_RECORD_VERSION, 1959
- tank_size, 1962
- tank_start, 1963
- tank_stop, 1963
- tank_store, 1963
- tree_alloc, 1964
- tree_count, 1964
- store
 - magma_keys_t, 87
 - tank.c, 1969
- store_message.c
 - mail_copy_message, 1172
 - mail_move_message, 1172
 - mail_store_message, 1173
 - mail_store_message_data, 1173
- str_printf
 - misc_pub.c, 1560
- str_tok_get_bl
 - core/parsers/parsers.h, 450
 - token.c, 477
- str_tok_get_count_bl
 - core/parsers/parsers.h, 450
 - token.c, 477
- string
 - api_lookup_t, 21
- stringer_t
 - strings.h, 543
- strings.h
 - __VA_NARG_N, 539
 - __VA_NARG_SEQ_N, 539
 - __VA_NARG__, 539
 - __attribute__, 543
 - BLOCK, 539
 - BLOCK_T, 543
 - block_t, 577
 - BLOCKBUF, 539
 - cmp_mt_mt, 543
 - CONSTANT, 539
 - CONSTANT_T, 543
 - constant_t, 577
 - CONTIGUOUS, 543
 - FOREIGNDATA, 543
 - HEAP, 543
 - ident_mt_mt, 544
 - JOINTED, 543
 - MANAGED, 539
 - MANAGED_T, 543
 - managed_t, 578
 - MANAGEDBUF, 539
 - MAPPED_T, 543
 - mapped_t, 578
 - mt_dupe, 544
 - mt_free, 545
 - mt_get_char, 545
 - mt_get_length, 545
 - mt_get_null, 546
 - mt_get_number, 546
 - mt_get_type, 546
 - mt_is_empty, 547
 - mt_is_number, 547
 - mt_set_type, 547
 - ns_alloc, 548
 - ns_append, 548
 - ns_cleanup, 540
 - ns_cleanup_variadic, 548
 - ns_dupe, 549
 - ns_empty, 549
 - ns_empty_out, 549
 - ns_free, 550
 - ns_import, 550
 - ns_length_get, 550
 - ns_length_int, 551
 - ns_populated, 540
 - ns_populated_variadic, 551
 - ns_wipe, 551
 - NULLER, 540
 - NULLER_T, 543
 - nuller_t, 578
 - pl_char_get, 552
 - pl_clone, 552
 - pl_data_get, 552
 - pl_empty, 552
 - pl_inc, 553
 - pl_init, 553
 - pl_length_get, 554
 - pl_length_int, 554
 - pl_null, 554
 - pl_set, 555
 - pl_starts_with_char, 555
 - PLACER, 540
 - PLACER_T, 543
 - placer_t, 578
 - SECURE, 543
 - st_alloc, 555
 - st_alloc_opts, 556
 - st_append, 541
 - st_append_opts, 556
 - st_append_out, 557
 - st_aprint, 557

- st_aprint_opts, 557
- st_avail_get, 557
- st_avail_set, 557
- st_char_get, 558
- st_cleanup, 541
- st_cleanup_variadic, 559
- st_copy_in, 559
- st_data_get, 560
- st_data_set, 560
- st_dupe, 561
- st_dupe_opts, 561
- st_empty, 542
- st_empty_out, 562
- st_empty_variadic, 562
- st_free, 563
- st_import, 564
- st_import_opts, 564
- st_info_allocator, 565
- st_info_layout, 565
- st_info_opts, 565
- st_info_type, 566
- st_length_get, 566
- st_length_int, 567
- st_length_set, 568
- st_merge, 542
- st_merge_opts, 569
- st_nullify, 569
- st_opt_get, 569
- st_opt_set, 570
- st_opt_test, 570
- st_output, 570
- st_populated, 542
- st_populated_variadic, 571
- st_quick, 571
- st_realloc, 571
- st_replace, 572
- st_set, 572
- st_sprint, 572
- st_swap, 572
- st_uchar_get, 573
- st_used_variadic, 573
- st_valid_append, 573
- st_valid_avail, 573
- st_valid_destination, 574
- st_valid_free, 574
- st_valid_joined, 574
- st_valid_opts, 575
- st_valid_placer, 575
- st_valid_tracked, 575
- st_vaprint, 542
- st_vaprint_opts, 576
- st_vsprint, 576
- st_wipe, 577
- st_write, 542
- st_write_variadic, 577
- STACK, 543
- stringer_t, 543
- va_narg, 543
- sub_explicit
 - mrec.h, 1765
- sub_relaxed
 - mrec.h, 1765
- sub_strict
 - mrec.h, 1765
- subdomain
 - dime_record_t, 39
- subject
 - mail_message_t, 104
 - smtp_message_t, 147
- SUBMISSION
 - engine/config/servers/servers.h, 753
- submission
 - smtp_session_t, 153
- submission_init
 - servers/smtp/smtp.h, 1008
 - smtp.c, 2155
- subnet_t, 156
 - address, 156
 - mask, 156
- suggested_eight_bit
 - smtp_session_t, 153
- suggested_length
 - smtp_session_t, 153
- symbol_t
 - providers.h, 1939
- symbols.c
 - __lzo_init_v2_d, 1984
 - ASN1_STRING_data_d, 1984
 - ASN1_STRING_TABLE_cleanup_d, 1984
 - BIO_free_all_d, 1985
 - BIO_free_d, 1985
 - BIO_new_fp_d, 1985
 - BIO_new_socket_d, 1985
 - BIO_sock_cleanup_d, 1985
 - BN_bn2bin_d, 1985
 - BN_bn2dec_d, 1985
 - BN_bn2hex_d, 1985
 - BN_bn2mpi_d, 1985
 - BN_cmp_d, 1985
 - BN_CTX_free_d, 1985
 - BN_CTX_new_d, 1985
 - BN_CTX_start_d, 1985
 - BN_free_d, 1985
 - BN_hex2bn_d, 1986
 - BN_mpi2bn_d, 1986
 - BN_num_bits_d, 1986
 - BUF_strlcat_d, 1986
 - BZ2_bzBuffToBuffCompress_d, 1986
 - BZ2_bzBuffToBuffDecompress_d, 1986
 - BZ2_bzlibVersion_d, 1986
 - cl_countsigs_d, 1986
 - cl_engine_compile_d, 1986
 - cl_engine_free_d, 1986
 - cl_engine_new_d, 1986

cl_engine_set_num_d, 1986
cl_engine_set_str_d, 1987
cl_init_d, 1987
cl_load_d, 1987
cl_retver_d, 1987
cl_scandesc_d, 1987
cl_shutdown_d, 1987
cl_statchkdir_d, 1987
cl_statfree_d, 1987
cl_statinidir_d, 1987
cl_strerror_d, 1987
COMP_zlib_cleanup_d, 1987
CONF_modules_unload_d, 1987
CRYPTO_cleanup_all_ex_data_d, 1988
CRYPTO_free_d, 1988
CRYPTO_num_locks_d, 1988
DH_check_d, 1988
DH_free_d, 1988
DH_new_d, 1988
dkim_body_d, 1988
dkim_chunk_d, 1988
dkim_close_d, 1988
dkim_eoh_d, 1988
dkim_eom_d, 1988
dkim_free_d, 1988
dkim_geterror_d, 1989
dkim_getresultstr_d, 1989
dkim_getsighdrx_d, 1989
dkim_header_d, 1989
dkim_init_d, 1989
dkim_libversion_d, 1989
dkim_mfree_d, 1989
dkim_sign_d, 1989
dkim_test_dns_put_d, 1989
dkim_verify_d, 1989
dspam_attach_d, 1989
dspam_create_d, 1989
dspam_destroy_d, 1989
dspam_detach_d, 1990
dspam_init_driver_d, 1990
dspam_process_d, 1990
dspam_shutdown_driver_d, 1990
dspam_version_d, 1990
EC_GROUP_clear_free_d, 1990
EC_GROUP_free_d, 1990
EC_GROUP_new_by_curve_name_d, 1990
EC_GROUP_precompute_mult_d, 1990
EC_KEY_check_key_d, 1990
EC_KEY_free_d, 1990
EC_KEY_generate_key_d, 1991
EC_KEY_get0_group_d, 1991
EC_KEY_get0_private_key_d, 1991
EC_KEY_get0_public_key_d, 1991
EC_KEY_new_by_curve_name_d, 1991
EC_KEY_new_d, 1991
EC_KEY_set_group_d, 1991
EC_KEY_set_private_key_d, 1991
EC_POINT_free_d, 1991
EC_POINT_new_d, 1991
ECDSA_SIG_free_d, 1992
ENGINE_cleanup_d, 1992
ERR_clear_error_d, 1992
ERR_free_strings_d, 1992
ERR_get_error_d, 1992
ERR_load_crypto_strings_d, 1992
ERR_peek_error_d, 1992
ERR_print_errors_fp_d, 1992
ERR_remove_thread_state_d, 1992
EVP_aes_256_cbc_d, 1992
EVP_aes_256_gcm_d, 1992
EVP_CIPHER_block_size_d, 1992
EVP_CIPHER_CTX_block_size_d, 1993
EVP_CIPHER_CTX_cleanup_d, 1993
EVP_CIPHER_CTX_free_d, 1993
EVP_CIPHER_CTX_init_d, 1993
EVP_CIPHER_CTX_iv_length_d, 1993
EVP_CIPHER_CTX_key_length_d, 1993
EVP_CIPHER_CTX_new_d, 1993
EVP_CIPHER_CTX_set_padding_d, 1993
EVP_CIPHER_flags_d, 1993
EVP_CIPHER_iv_length_d, 1993
EVP_CIPHER_key_length_d, 1993
EVP_CIPHER_nid_d, 1994
EVP_cleanup_d, 1994
EVP_DigestInit_d, 1994
EVP_get_cipherbyname_d, 1994
EVP_get_digestbyname_d, 1994
EVP_md4_d, 1994
EVP_md5_d, 1994
EVP_MD_CTX_cleanup_d, 1994
EVP_MD_CTX_init_d, 1994
EVP_MD_size_d, 1994
EVP_MD_type_d, 1994
EVP_PKEY_new_d, 1994
EVP_PKEY_set1_RSA_d, 1995
EVP_ripemd160_d, 1995
EVP_sha1_d, 1995
EVP_sha224_d, 1995
EVP_sha256_d, 1995
EVP_sha384_d, 1995
EVP_sha512_d, 1995
EVP_sha_d, 1995
FT_Done_FreeType_d, 1995
FT_Init_FreeType_d, 1995
FT_Library_Version_d, 1995
gdFree_d, 1995
gdImageColorResolve_d, 1996
gdImageCreate_d, 1996
gdImageDestroy_d, 1996
gdImageGifPtr_d, 1996
gdImageJpegPtr_d, 1996
gdImageSetPixel_d, 1996
gdImageStringFT_d, 1996
gdVersionString_d, 1996

HMAC_CTX_cleanup_d, 1996
 HMAC_CTX_init_d, 1996
 i2d_ECPrivateKey_d, 1996
 i2d_OCSP_CERTID_d, 1997
 i2d_X509_d, 1997
 i2o_ECPublicKey_d, 1997
 jpeg_version_d, 1997
 lib_load, 1984
 lib_unload, 1984
 lt_dllexit_d, 1997
 lzo1x_l_compress_d, 1997
 lzo1x_decompress_safe_d, 1997
 lzo_adler32_d, 1997
 lzo_version_string_d, 1997
 memcached_add_d, 1997
 memcached_append_d, 1997
 memcached_behavior_set_d, 1997
 memcached_cas_d, 1997
 memcached_create_d, 1998
 memcached_decrement_d, 1998
 memcached_decrement_with_initial_d, 1998
 memcached_delete_d, 1998
 memcached_flush_d, 1998
 memcached_free_d, 1998
 memcached_get_d, 1998
 memcached_increment_d, 1998
 memcached_increment_with_initial_d, 1998
 memcached_lib_version_d, 1998
 memcached_prepend_d, 1998
 memcached_replace_d, 1999
 memcached_server_add_with_weight_d, 1999
 memcached_set_d, 1999
 memcached_strerror_d, 1999
 my_once_free_d, 1999
 mysql_affected_rows_d, 1999
 mysql_character_set_name_d, 1999
 mysql_close_d, 1999
 mysql_embedded_d, 1999
 mysql_errno_d, 1999
 mysql_error_d, 1999
 mysql_escape_string_d, 1999
 mysql_fetch_field_d, 2000
 mysql_fetch_row_d, 2000
 mysql_free_result_d, 2000
 mysql_get_client_version_d, 2000
 mysql_get_server_info_d, 2000
 mysql_init_d, 2000
 mysql_insert_id_d, 2000
 mysql_num_fields_d, 2000
 mysql_num_rows_d, 2000
 mysql_options_d, 2000
 mysql_ping_d, 2000
 mysql_real_connect_d, 2000
 mysql_real_query_d, 2000
 mysql_server_end_d, 2001
 mysql_server_init_d, 2001
 mysql_set_character_set_d, 2001
 mysql_stmt_affected_rows_d, 2001
 mysql_stmt_attr_set_d, 2001
 mysql_stmt_bind_param_d, 2001
 mysql_stmt_bind_result_d, 2001
 mysql_stmt_close_d, 2001
 mysql_stmt_errno_d, 2001
 mysql_stmt_error_d, 2001
 mysql_stmt_execute_d, 2001
 mysql_stmt_fetch_d, 2001
 mysql_stmt_free_result_d, 2002
 mysql_stmt_init_d, 2002
 mysql_stmt_insert_id_d, 2002
 mysql_stmt_num_rows_d, 2002
 mysql_stmt_prepare_d, 2002
 mysql_stmt_reset_d, 2002
 mysql_stmt_result_metadata_d, 2002
 mysql_stmt_store_result_d, 2002
 mysql_store_result_d, 2002
 mysql_thread_end_d, 2002
 mysql_thread_id_d, 2002
 mysql_thread_init_d, 2002
 mysql_thread_safe_d, 2003
 OBJ_cleanup_d, 2003
 OBJ_NAME_cleanup_d, 2003
 OBJ_nid2sn_d, 2003
 OCSP_BASICRESP_free_d, 2003
 OCSP_check_nonce_d, 2003
 OCSP_REQ_CTX_set1_req_d, 2003
 OCSP_request_add0_id_d, 2003
 OCSP_REQUEST_free_d, 2003
 OCSP_REQUEST_new_d, 2003
 OCSP_RESPONSE_free_d, 2003
 OCSP_response_get1_basic_d, 2003
 OCSP_response_status_d, 2004
 OCSP_response_status_str_d, 2004
 OPENSSL_add_all_algorithms_noconf_d, 2004
 RAND_bytes_d, 2004
 RAND_cleanup_d, 2004
 RAND_load_file_d, 2004
 RAND_status_d, 2004
 RSA_free_d, 2004
 RSA_new_d, 2004
 RSAPublicKey_dup_d, 2004
 SHA1_Final_d, 2004
 SHA1_Init_d, 2004
 SHA1_Update_d, 2005
 SHA256_Final_d, 2005
 SHA256_Init_d, 2005
 SHA512_Final_d, 2005
 SHA512_Init_d, 2005
 sk_num_d, 2005
 sk_pop_d, 2005
 sk_pop_free_d, 2005
 sk_value_d, 2005
 SSL_accept_d, 2005
 SSL_CIPHER_get_name_d, 2005
 SSL_CIPHER_get_version_d, 2005

- SSL_connect_d, 2006
- SSL_ctrl_d, 2006
- SSL_CTX_callback_ctrl_d, 2006
- SSL_CTX_check_private_key_d, 2006
- SSL_CTX_free_d, 2006
- SSL_CTX_new_d, 2006
- SSL_CTX_set_cipher_list_d, 2006
- SSL_do_handshake_d, 2006
- SSL_free_d, 2006
- SSL_get_current_cipher_d, 2006
- SSL_get_error_d, 2006
- SSL_get_fd_d, 2006
- SSL_get_peer_certificate_d, 2006
- SSL_get_read_ahead_d, 2007
- SSL_get_rfd_d, 2007
- SSL_get_shutdown_d, 2007
- SSL_get_version_d, 2007
- SSL_get_wbio_d, 2007
- SSL_library_init_d, 2007
- SSL_load_error_strings_d, 2007
- SSL_new_d, 2007
- SSL_peek_d, 2007
- SSL_pending_d, 2007
- SSL_read_d, 2007
- SSL_set_accept_state_d, 2007
- SSL_set_bio_d, 2007
- SSL_set_connect_state_d, 2007
- SSL_set_fd_d, 2008
- SSL_set_read_ahead_d, 2008
- SSL_shutdown_d, 2008
- SSL_version_str_d, 2008
- SSL_want_d, 2008
- SSL_write_d, 2008
- SSLeay_version_d, 2008
- SSLv23_client_method_d, 2008
- SSLv23_server_method_d, 2008
- STACK_OF, 1984
- TLSv1_server_method_d, 2008
- X509_check_issued_d, 2008
- X509_email_free_d, 2008
- X509_get1_ocsp_d, 2008
- X509_get_ext_count_d, 2008
- X509_get_ext_d, 2009
- X509_get_subject_name_d, 2009
- X509_LOOKUP_file_d, 2009
- X509_NAME_online_d, 2009
- X509_STORE_CTX_free_d, 2009
- X509_STORE_CTX_get_error_d, 2009
- X509_STORE_CTX_get_error_depth_d, 2009
- X509_STORE_CTX_new_d, 2009
- X509_STORE_free_d, 2009
- X509_STORE_new_d, 2009
- X509_verify_cert_d, 2009
- X509_verify_cert_error_string_d, 2009
- symbols.h
 - _lzo_init_v2_d, 2023
 - ASN1_GENERALIZEDTIME_print_d, 2023
 - ASN1_INTEGER_to_BN_d, 2023
 - ASN1_STRING_data_d, 2023
 - ASN1_STRING_TABLE_cleanup_d, 2024
 - BIO_free_all_d, 2024
 - BIO_free_d, 2024
 - BIO_new_fp_d, 2024
 - BIO_new_socket_d, 2024
 - BIO_sock_cleanup_d, 2024
 - BIO_vprintf_d, 2024
 - BN_bin2bn_d, 2024
 - BN_bn2bin_d, 2024
 - BN_bn2dec_d, 2024
 - BN_bn2hex_d, 2024
 - BN_bn2mpi_d, 2024
 - BN_cmp_d, 2025
 - BN_CTX_free_d, 2025
 - BN_CTX_new_d, 2025
 - BN_CTX_start_d, 2025
 - BN_free_d, 2025
 - BN_hex2bn_d, 2025
 - BN_mpi2bn_d, 2025
 - BN_num_bits_d, 2025
 - BUF_strlcat_d, 2025
 - BZ2_bzBuffToBuffCompress_d, 2025
 - BZ2_bzBuffToBuffDecompress_d, 2025
 - BZ2_bzlibVersion_d, 2025
 - cl_countsigs_d, 2025
 - cl_engine_compile_d, 2026
 - cl_engine_free_d, 2026
 - cl_engine_new_d, 2026
 - cl_engine_set_num_d, 2026
 - cl_engine_set_str_d, 2026
 - cl_init_d, 2026
 - cl_load_d, 2026
 - cl_retver_d, 2026
 - cl_scandesc_d, 2026
 - cl_shutdown_d, 2026
 - cl_statchkdir_d, 2026
 - cl_statfree_d, 2026
 - cl_statinidir_d, 2027
 - cl_strerror_d, 2027
 - COMP_zlib_cleanup_d, 2027
 - compress2_d, 2027
 - compressBound_d, 2027
 - CONF_modules_unload_d, 2027
 - CONFIG_DEFAULT, 2023
 - CRYPTO_cleanup_all_ex_data_d, 2027
 - CRYPTO_free_d, 2027
 - CRYPTO_num_locks_d, 2027
 - CRYPTO_set_id_callback_d, 2027
 - CRYPTO_set_locked_mem_functions_d, 2027
 - CRYPTO_set_locking_callback_d, 2027
 - CRYPTO_set_mem_functions_d, 2028
 - d2i_ECDSA_SIG_d, 2028
 - d2i_ECPrivateKey_d, 2028
 - d2i_OCSP_RESPONSE_d, 2028
 - DH_check_d, 2028

DH_free_d, 2028
 DH_generate_parameters_ex_d, 2028
 DH_new_d, 2028
 dkim_body_d, 2028
 dkim_chunk_d, 2028
 dkim_close_d, 2028
 dkim_eoh_d, 2028
 dkim_eom_d, 2029
 dkim_free_d, 2029
 dkim_geterror_d, 2029
 dkim_getresultstr_d, 2029
 dkim_getsighdrx_d, 2029
 dkim_header_d, 2029
 dkim_init_d, 2029
 dkim_libversion_d, 2029
 dkim_mfree_d, 2029
 dkim_sign_d, 2029
 dkim_test_dns_put_d, 2029
 dkim_verify_d, 2029
 dspam_attach_d, 2029
 dspam_create_d, 2030
 dspam_destroy_d, 2030
 dspam_detach_d, 2030
 dspam_init_driver_d, 2030
 dspam_process_d, 2030
 dspam_shutdown_driver_d, 2030
 dspam_version_d, 2030
 EC_GROUP_clear_free_d, 2030
 EC_GROUP_free_d, 2030
 EC_GROUP_new_by_curve_name_d, 2030
 EC_GROUP_precompute_mult_d, 2030
 EC_GROUP_set_point_conversion_form_d, 2031
 EC_KEY_check_key_d, 2031
 EC_KEY_free_d, 2031
 EC_KEY_generate_key_d, 2031
 EC_KEY_get0_group_d, 2031
 EC_KEY_get0_private_key_d, 2031
 EC_KEY_get0_public_key_d, 2031
 EC_KEY_get_conv_form_d, 2031
 EC_KEY_new_by_curve_name_d, 2031
 EC_KEY_new_d, 2031
 EC_KEY_set_conv_form_d, 2032
 EC_KEY_set_group_d, 2032
 EC_KEY_set_private_key_d, 2032
 EC_KEY_set_public_key_d, 2032
 EC_POINT_cmp_d, 2032
 EC_POINT_free_d, 2032
 EC_POINT_hex2point_d, 2032
 EC_POINT_mul_d, 2032
 EC_POINT_new_d, 2032
 EC_POINT_oct2point_d, 2032
 EC_POINT_point2hex_d, 2032
 EC_POINT_point2oct_d, 2032
 ECDH_compute_key_d, 2033
 ECDSA_do_sign_d, 2033
 ECDSA_do_verify_d, 2033
 ECDSA_SIG_free_d, 2033
 ED25519_keypair_d, 2033
 ED25519_keypair_from_seed_d, 2033
 ED25519_sign_d, 2033
 ED25519_verify_d, 2033
 ENGINE_cleanup_d, 2033
 ERR_clear_error_d, 2033
 ERR_error_string_n_d, 2033
 ERR_free_strings_d, 2034
 ERR_get_error_d, 2034
 ERR_load_crypto_strings_d, 2034
 ERR_peek_error_d, 2034
 ERR_peek_error_line_data_d, 2034
 ERR_print_errors_fp_d, 2034
 ERR_put_error_d, 2034
 ERR_remove_thread_state_d, 2034
 EVP_aes_256_cbc_d, 2034
 EVP_aes_256_gcm_d, 2034
 EVP_CIPHER_block_size_d, 2034
 EVP_CIPHER_CTX_block_size_d, 2034
 EVP_CIPHER_CTX_cleanup_d, 2034
 EVP_CIPHER_CTX_ctrl_d, 2035
 EVP_CIPHER_CTX_flags_d, 2035
 EVP_CIPHER_CTX_free_d, 2035
 EVP_CIPHER_CTX_init_d, 2035
 EVP_CIPHER_CTX_iv_length_d, 2035
 EVP_CIPHER_CTX_key_length_d, 2035
 EVP_CIPHER_CTX_new_d, 2035
 EVP_CIPHER_CTX_set_padding_d, 2035
 EVP_CIPHER_flags_d, 2035
 EVP_CIPHER_iv_length_d, 2035
 EVP_CIPHER_key_length_d, 2035
 EVP_CIPHER_nid_d, 2036
 EVP_cleanup_d, 2036
 EVP_DecryptFinal_ex_d, 2036
 EVP_DecryptInit_ex_d, 2036
 EVP_DecryptUpdate_d, 2036
 EVP_Digest_d, 2036
 EVP_DigestFinal_d, 2036
 EVP_DigestFinal_ex_d, 2036
 EVP_DigestInit_d, 2036
 EVP_DigestInit_ex_d, 2036
 EVP_DigestUpdate_d, 2036
 EVP_EncryptFinal_ex_d, 2036
 EVP_EncryptInit_ex_d, 2037
 EVP_EncryptUpdate_d, 2037
 EVP_get_cipherbyname_d, 2037
 EVP_get_digestbyname_d, 2037
 EVP_md4_d, 2037
 EVP_md5_d, 2037
 EVP_MD_CTX_cleanup_d, 2037
 EVP_MD_CTX_init_d, 2037
 EVP_MD_size_d, 2037
 EVP_MD_type_d, 2037
 EVP_PKEY_new_d, 2037
 EVP_PKEY_set1_RSA_d, 2038
 EVP_ripemd160_d, 2038
 EVP_sha1_d, 2038

EVP_sha224_d, 2038
EVP_sha256_d, 2038
EVP_sha384_d, 2038
EVP_sha512_d, 2038
EVP_sha_d, 2038
EVP_VerifyFinal_d, 2038
FT_Done_FreeType_d, 2038
FT_Init_FreeType_d, 2038
FT_Library_Version_d, 2038
gdFree_d, 2039
gdImageColorResolve_d, 2039
gdImageCreate_d, 2039
gdImageDestroy_d, 2039
gdImageGifPtr_d, 2039
gdImageJpegPtr_d, 2039
gdImageSetPixel_d, 2039
gdImageStringFT_d, 2039
gdVersionString_d, 2039
HMAC_CTX_cleanup_d, 2039
HMAC_CTX_init_d, 2039
HMAC_Final_d, 2039
HMAC_Init_ex_d, 2040
HMAC_Update_d, 2040
i2d_ECDSA_SIG_d, 2040
i2d_ECPrivateKey_d, 2040
i2d_OCSP_CERTID_d, 2040
i2d_OCSP_RESPONSE_d, 2040
i2d_X509_d, 2040
i2o_ECPublicKey_d, 2040
jansson_version_d, 2040
jpeg_version_d, 2040
json_array_append_d, 2040
json_array_append_new_d, 2041
json_array_clear_d, 2041
json_array_d, 2041
json_array_extend_d, 2041
json_array_get_d, 2041
json_array_insert_d, 2041
json_array_insert_new_d, 2041
json_array_remove_d, 2041
json_array_set_d, 2041
json_array_set_new_d, 2041
json_array_size_d, 2041
json_copy_d, 2041
json_decref_d, 2041
json_deep_copy_d, 2041
json_delete_d, 2042
json_dump_file_d, 2042
json_dumpf_d, 2042
json_dumps_d, 2042
json_equal_d, 2042
json_false_d, 2042
json_incref_d, 2042
json_integer_d, 2042
json_integer_set_d, 2042
json_integer_value_d, 2042
json_load_file_d, 2042
json_loadf_d, 2042
json_loads_d, 2042
json_null_d, 2042
json_number_value_d, 2042
json_object_clear_d, 2042
json_object_d, 2042
json_object_del_d, 2042
json_object_get_d, 2043
json_object_iter_at_d, 2043
json_object_iter_d, 2043
json_object_iter_key_d, 2043
json_object_iter_next_d, 2043
json_object_iter_set_d, 2043
json_object_iter_set_new_d, 2043
json_object_iter_value_d, 2043
json_object_set_d, 2043
json_object_set_new_d, 2043
json_object_set_new_nocheck_d, 2043
json_object_set_nocheck_d, 2043
json_object_size_d, 2043
json_object_update_d, 2043
json_pack_d, 2044
json_pack_ex_d, 2044
json_real_d, 2044
json_real_set_d, 2044
json_real_value_d, 2044
json_set_alloc_funcs_d, 2044
json_string_d, 2044
json_string_nocheck_d, 2044
json_string_set_d, 2044
json_string_set_nocheck_d, 2044
json_string_value_d, 2044
json_true_d, 2044
json_type_string_d, 2044
json_unpack_d, 2044
json_unpack_ex_d, 2044
json_vpack_ex_d, 2045
json_vunpack_ex_d, 2045
lint, 2023
LOGDIR, 2023
lt_dlexit_d, 2045
lzo1x_1_compress_d, 2045
lzo1x_decompress_safe_d, 2045
lzo_adler32_d, 2045
lzo_version_string_d, 2045
M_BIND, 2023
memcached_add_d, 2045
memcached_append_d, 2045
memcached_behavior_set_d, 2045
memcached_cas_d, 2045
memcached_create_d, 2045
memcached_decrement_d, 2045
memcached_decrement_with_initial_d, 2046
memcached_delete_d, 2046
memcached_flush_d, 2046
memcached_free_d, 2046
memcached_get_d, 2046

memcached_increment_d, 2046
 memcached_increment_with_initial_d, 2046
 memcached_lib_version_d, 2046
 memcached_prepend_d, 2046
 memcached_replace_d, 2046
 memcached_server_add_with_weight_d, 2046
 memcached_set_d, 2046
 memcached_strerror_d, 2047
 my_once_free_d, 2047
 mysql_affected_rows_d, 2047
 mysql_character_set_name_d, 2047
 mysql_close_d, 2047
 mysql_embedded_d, 2047
 mysql_errno_d, 2047
 mysql_error_d, 2047
 mysql_escape_string_d, 2047
 mysql_fetch_field_d, 2047
 mysql_fetch_row_d, 2047
 mysql_free_result_d, 2047
 mysql_get_client_version_d, 2047
 mysql_get_server_info_d, 2048
 mysql_init_d, 2048
 mysql_insert_id_d, 2048
 mysql_num_fields_d, 2048
 mysql_num_rows_d, 2048
 mysql_options_d, 2048
 mysql_ping_d, 2048
 mysql_real_connect_d, 2048
 mysql_real_query_d, 2048
 mysql_server_end_d, 2048
 mysql_server_init_d, 2048
 mysql_set_character_set_d, 2048
 mysql_stmt_affected_rows_d, 2049
 mysql_stmt_attr_set_d, 2049
 mysql_stmt_bind_param_d, 2049
 mysql_stmt_bind_result_d, 2049
 mysql_stmt_close_d, 2049
 mysql_stmt_errno_d, 2049
 mysql_stmt_error_d, 2049
 mysql_stmt_execute_d, 2049
 mysql_stmt_fetch_d, 2049
 mysql_stmt_free_result_d, 2049
 mysql_stmt_init_d, 2049
 mysql_stmt_insert_id_d, 2049
 mysql_stmt_num_rows_d, 2049
 mysql_stmt_prepare_d, 2050
 mysql_stmt_reset_d, 2050
 mysql_stmt_result_metadata_d, 2050
 mysql_stmt_store_result_d, 2050
 mysql_store_result_d, 2050
 mysql_thread_end_d, 2050
 mysql_thread_id_d, 2050
 mysql_thread_init_d, 2050
 mysql_thread_safe_d, 2050
 o2i_ECPrivateKey_d, 2050
 OBJ_cleanup_d, 2050
 OBJ_NAME_cleanup_d, 2050
 OBJ_nid2sn_d, 2051
 OCSP_basic_verify_d, 2051
 OCSP_BASICRESP_free_d, 2051
 OCSP_cert_to_id_d, 2051
 OCSP_check_nonce_d, 2051
 OCSP_check_validity_d, 2051
 OCSP_parse_url_d, 2051
 OCSP_REQ_CTX_add1_header_d, 2051
 OCSP_REQ_CTX_set1_req_d, 2051
 OCSP_request_add0_id_d, 2051
 OCSP_request_add1_nonce_d, 2051
 OCSP_REQUEST_free_d, 2051
 OCSP_REQUEST_new_d, 2052
 OCSP_REQUEST_print_d, 2052
 OCSP_resp_find_status_d, 2052
 OCSP_RESPONSE_free_d, 2052
 OCSP_response_get1_basic_d, 2052
 OCSP_RESPONSE_print_d, 2052
 OCSP_response_status_d, 2052
 OCSP_response_status_str_d, 2052
 OCSP_sendreq_nbio_d, 2052
 OCSP_sendreq_new_d, 2052
 OPENSSL_add_all_algorithms_noconf_d, 2052
 png_access_version_number_d, 2052
 RAND_bytes_d, 2053
 RAND_cleanup_d, 2053
 RAND_load_file_d, 2053
 RAND_status_d, 2053
 RSA_free_d, 2053
 RSA_new_d, 2053
 RSAPublicKey_dup_d, 2053
 SHA1_Final_d, 2053
 SHA1_Init_d, 2053
 SHA1_Update_d, 2053
 SHA256_Final_d, 2053
 SHA256_Init_d, 2054
 SHA256_Update_d, 2054
 SHA512_d, 2054
 SHA512_Final_d, 2054
 SHA512_Init_d, 2054
 SHA512_Update_d, 2054
 sk_num_d, 2054
 sk_pop_d, 2054
 sk_pop_free_d, 2054
 sk_value_d, 2054
 SPF_dns_zone_add_str_d, 2054
 SPF_dns_zone_new_d, 2054
 SPF_get_lib_version_d, 2054
 SPF_request_free_d, 2054
 SPF_request_new_d, 2055
 SPF_request_query_mailfrom_d, 2055
 SPF_request_set_env_from_d, 2055
 SPF_request_set_helo_dom_d, 2055
 SPF_request_set_ipv4_d, 2055
 SPF_request_set_ipv6_d, 2055
 SPF_response_free_d, 2055
 SPF_response_reason_d, 2055

SPF_response_result_d, 2055
SPF_server_free_d, 2055
SPF_server_new_d, 2055
SPF_strerror_d, 2055
SPF_strerror_reason_d, 2056
SPF_strerror_result_d, 2056
SSL_accept_d, 2056
SSL_CIPHER_get_bits_d, 2056
SSL_CIPHER_get_name_d, 2056
SSL_CIPHER_get_version_d, 2056
SSL_connect_d, 2056
SSL_ctrl_d, 2056
SSL_CTX_callback_ctrl_d, 2056
SSL_CTX_check_private_key_d, 2056
SSL_CTX_ctrl_d, 2056
SSL_CTX_free_d, 2056
SSL_CTX_load_verify_locations_d, 2056
SSL_CTX_new_d, 2057
SSL_CTX_set_cipher_list_d, 2057
SSL_CTX_set_tmp_dh_callback_d, 2057
SSL_CTX_set_tmp_ecdh_callback_d, 2057
SSL_CTX_set_verify_d, 2057
SSL_CTX_use_certificate_chain_file_d, 2057
SSL_CTX_use_PrivateKey_file_d, 2057
SSL_do_handshake_d, 2057
SSL_free_d, 2057
SSL_get_current_cipher_d, 2057
SSL_get_error_d, 2057
SSL_get_fd_d, 2057
SSL_get_peer_cert_chain_d, 2057
SSL_get_peer_certificate_d, 2058
SSL_get_read_ahead_d, 2058
SSL_get_rfd_d, 2058
SSL_get_shutdown_d, 2058
SSL_get_version_d, 2058
SSL_get_wbio_d, 2058
SSL_library_init_d, 2058
SSL_load_error_strings_d, 2058
SSL_new_d, 2058
SSL_peek_d, 2058
SSL_pending_d, 2058
SSL_read_d, 2058
SSL_set_accept_state_d, 2058
SSL_set_bio_d, 2058
SSL_set_connect_state_d, 2059
SSL_set_fd_d, 2059
SSL_set_read_ahead_d, 2059
SSL_shutdown_d, 2059
SSL_version_str_d, 2059
SSL_want_d, 2059
SSL_write_d, 2059
SSLeay_version_d, 2059
SSLv23_client_method_d, 2059
SSLv23_server_method_d, 2059
STACK_OF, 2023
tcfree_d, 2059
tchdbclose_d, 2059
tchdbdefrag_d, 2059
tchdbdel_d, 2059
tchdbdecode_d, 2060
tchdberrmsg_d, 2060
tchdbfsiz_d, 2060
tchdbget_d, 2060
tchdbnew_d, 2060
tchdbopen_d, 2060
tchdboptimize_d, 2060
tchdbout_d, 2060
tchdbpath_d, 2060
tchdbputasync_d, 2060
tchdbnum_d, 2060
tchdbsetdfunit_d, 2060
tchdbsetmutex_d, 2061
tchdbsync_d, 2061
tchdbtune_d, 2061
tclistdel_d, 2061
tclistnum_d, 2061
tclistval_d, 2061
tcndbdel_d, 2061
tcndbdup_d, 2061
tcndbfwmkeys_d, 2061
tcndbget3_d, 2061
tcndbget_d, 2061
tcndbgetboth_d, 2061
tcndbiterinit_d, 2061
tcndbiternext2_d, 2061
tcndbnew2_d, 2062
tcndbout_d, 2062
tcndbputkeep_d, 2062
tcndbnum_d, 2062
tctreeclear_d, 2062
tctreekeys_d, 2062
tctreevals_d, 2062
tcversion_d, 2062
TLSv1_server_method_d, 2062
uncompress_d, 2062
utf8proc_category_d, 2062
utf8proc_category_string_d, 2062
utf8proc_errmsg_d, 2062
utf8proc_get_property_d, 2062
utf8proc_iterate_d, 2063
utf8proc_version_d, 2063
X509_check_host_d, 2063
X509_check_issued_d, 2063
X509_email_free_d, 2063
X509_get1_ocsp_d, 2063
X509_get_ext_count_d, 2063
X509_get_ext_d, 2063
X509_get_subject_name_d, 2063
X509_LOOKUP_file_d, 2063
X509_NAME_ENTRY_get_data_d, 2063
X509_NAME_get_entry_d, 2063
X509_NAME_get_index_by_NID_d, 2063
X509_NAME_get_text_by_NID_d, 2064
X509_NAME_online_d, 2064

- X509_STORE_add_lookup_d, 2064
- X509_STORE_CTX_free_d, 2064
- X509_STORE_CTX_get_current_cert_d, 2064
- X509_STORE_CTX_get_error_d, 2064
- X509_STORE_CTX_get_error_depth_d, 2064
- X509_STORE_CTX_init_d, 2064
- X509_STORE_CTX_new_d, 2064
- X509_STORE_CTX_set_chain_d, 2064
- X509_STORE_free_d, 2064
- X509_STORE_load_locations_d, 2064
- X509_STORE_new_d, 2065
- X509_STORE_set_flags_d, 2065
- X509_verify_cert_d, 2065
- X509_verify_cert_error_string_d, 2065
- xmlAddSibling_d, 2065
- xmlBufferContent_d, 2065
- xmlBufferCreate_d, 2065
- xmlBufferFree_d, 2065
- xmlBufferLength_d, 2065
- xmlCleanupGlobals_d, 2065
- xmlCleanupParser_d, 2065
- xmlCtxtReadMemory_d, 2065
- xmlDocDumpFormatMemory_d, 2066
- xmlEncodeEntitiesReentrant_d, 2066
- xmlFreeDoc_d, 2066
- xmlFreeNode_d, 2066
- xmlFreeParserCtxt_d, 2066
- xmlInitParser_d, 2066
- xmlMemoryDump_d, 2066
- xmlNewNode_d, 2066
- xmlNewParserCtxt_d, 2066
- xmlNodeBufGetContent_d, 2066
- xmlNodeSetContent_d, 2066
- xmlParserVersion_d, 2066
- xmlSetProp_d, 2067
- xmlXPathEvalExpression_d, 2067
- xmlXPathFreeContext_d, 2067
- xmlXPathFreeObject_d, 2067
- xmlXPathNewContext_d, 2067
- xmlXPathRegisterNs_d, 2067
- zlibVersion_d, 2067
- symmetric_decrypt
 - deprecated.h, 1508
 - deprecated/symmetric.c, 1434
- symmetric_encrypt
 - deprecated.h, 1508
 - deprecated/symmetric.c, 1434
- symmetric_key
 - deprecated.h, 1508
 - deprecated/symmetric.c, 1435
- symmetric_vector
 - deprecated.h, 1509
 - deprecated/symmetric.c, 1435
- syndicates
 - dime_record_t, 39
- system
 - magma_t, 101
 - tank.c, 1969
- system.c
 - system_change_root_directory, 774
 - system_fork_daemon, 774
 - system_init_core_dumps, 775
 - system_init_impersonation, 775
 - system_init_resource_limits, 775
 - system_init_umask, 775
 - system_ulimit_cur, 776
 - system_ulimit_max, 776
- system_change_root_directory
 - context.h, 766
 - system.c, 774
- system_fork_daemon
 - context.h, 766
 - system.c, 774
- system_init_core_dumps
 - context.h, 767
 - system.c, 775
- system_init_impersonation
 - context.h, 767
 - system.c, 775
- system_init_resource_limits
 - context.h, 767
 - system.c, 775
- system_init_umask
 - context.h, 768
 - system.c, 775
- system_ulimit_cur
 - context.h, 768
 - system.c, 776
- system_ulimit_max
 - context.h, 768
 - system.c, 776
- T
 - ge_p1p1, 70
 - ge_p3, 72
- t
 - ge25519_p1p1_t, 66
 - ge25519_t, 68
- T2d
 - ge_cached, 69
- t2d
 - ge25519_niels_t, 65
 - ge25519_pniels_t, 67
- T_DNSKEY
 - dns.h, 1750
- T_DS
 - dns.h, 1750
- T_RRSIG
 - dns.h, 1750
- table_t
 - database.h, 1445
- tag
 - meta_stats_tag_t, 110
- tank

- magma_t, [101](#)
- tank.c
 - store, [1969](#)
 - system, [1969](#)
 - tank_close, [1966](#)
 - tank_count, [1967](#)
 - tank_cycle, [1967](#)
 - tank_delete, [1967](#)
 - tank_load, [1967](#)
 - tank_maintain, [1968](#)
 - tank_open, [1968](#)
 - tank_size, [1968](#)
 - tank_start, [1968](#)
 - tank_stop, [1969](#)
 - tank_store, [1969](#)
 - tanks, [1970](#)
 - tanks_lock, [1970](#)
 - tanks_next, [1970](#)
 - tanks_num, [1970](#)
 - tuner, [1970](#)
- TANK_COMPRESS_BZIP
 - storage.h, [1959](#)
- TANK_COMPRESS_LZO
 - storage.h, [1959](#)
- TANK_COMPRESS_ZLIB
 - storage.h, [1959](#)
- tank_close
 - tank.c, [1966](#)
- tank_count
 - storage.h, [1960](#)
 - tank.c, [1967](#)
- tank_cycle
 - storage.h, [1960](#)
 - tank.c, [1967](#)
- tank_delete
 - storage.h, [1961](#)
 - tank.c, [1967](#)
- tank_delete_object
 - providers/storage/data.c, [499](#)
 - storage.h, [1961](#)
- TANK_ENTRY_VERSION
 - storage.h, [1959](#)
- TANK_FLAGS_E
 - storage.h, [1965](#)
- tank_insert_object
 - providers/storage/data.c, [499](#)
 - storage.h, [1961](#)
- tank_load
 - storage.h, [1962](#)
 - tank.c, [1967](#)
- tank_maintain
 - storage.h, [1962](#)
 - tank.c, [1968](#)
- tank_open
 - tank.c, [1968](#)
- TANK_RECORD_VERSION
 - storage.h, [1959](#)
- tank_size
 - storage.h, [1962](#)
 - tank.c, [1968](#)
- tank_start
 - storage.h, [1963](#)
 - tank.c, [1968](#)
- tank_stop
 - storage.h, [1963](#)
 - tank.c, [1969](#)
- tank_store
 - storage.h, [1963](#)
 - tank.c, [1969](#)
- tanks
 - tank.c, [1970](#)
- tanks_lock
 - tank.c, [1970](#)
- tanks_next
 - tank.c, [1970](#)
- tanks_num
 - tank.c, [1970](#)
- tcfree_d
 - symbols.h, [2059](#)
- tchdbclose_d
 - symbols.h, [2059](#)
- tchdbdefrag_d
 - symbols.h, [2059](#)
- tchdbdel_d
 - symbols.h, [2059](#)
- tchdbdecode_d
 - symbols.h, [2060](#)
- tchdberrmsg_d
 - symbols.h, [2060](#)
- tchdbfsiz_d
 - symbols.h, [2060](#)
- tchdbget_d
 - symbols.h, [2060](#)
- tchdbnew_d
 - symbols.h, [2060](#)
- tchdbopen_d
 - symbols.h, [2060](#)
- tchdboptimize_d
 - symbols.h, [2060](#)
- tchdbout_d
 - symbols.h, [2060](#)
- tchdbpath_d
 - symbols.h, [2060](#)
- tchdbputasync_d
 - symbols.h, [2060](#)
- tchdbnum_d
 - symbols.h, [2060](#)
- tchdbsetdfunit_d
 - symbols.h, [2060](#)
- tchdbsetmutex_d
 - symbols.h, [2061](#)
- tchdbsync_d
 - symbols.h, [2061](#)
- tchdbtune_d

- symbols.h, 2061
- tcldel_d
 - symbols.h, 2061
- tcldnum_d
 - symbols.h, 2061
- tcldval_d
 - symbols.h, 2061
- tcndbdel_d
 - symbols.h, 2061
- tcndbdup_d
 - symbols.h, 2061
- tcndbfwmkeys_d
 - symbols.h, 2061
- tcndbget3_d
 - symbols.h, 2061
- tcndbget_d
 - symbols.h, 2061
- tcndbgetboth_d
 - symbols.h, 2061
- tcndbiterinit_d
 - symbols.h, 2061
- tcndbiternext2_d
 - symbols.h, 2061
- tcndbnew2_d
 - symbols.h, 2062
- tcndbout_d
 - symbols.h, 2062
- tcndbputkeep_d
 - symbols.h, 2062
- tcndbrnum_d
 - symbols.h, 2062
- tcp.c
 - tcp_addr_ip, 323
 - tcp_addr_st, 323
 - tcp_continue, 323
 - tcp_error, 324
 - tcp_read, 324
 - tcp_status, 324
 - tcp_wait, 324
 - tcp_write, 325
- TCP_PORT
 - engine/config/servers/servers.h, 753
- tcp_addr_ip
 - host.h, 304
 - tcp.c, 323
- tcp_addr_st
 - host.h, 304
 - tcp.c, 323
- tcp_continue
 - host.h, 304
 - tcp.c, 323
- tcp_error
 - host.h, 304
 - tcp.c, 324
- tcp_port
 - server_config_t, 128
- tcp_read
 - host.h, 305
 - tcp.c, 324
- tcp_status
 - host.h, 305
 - tcp.c, 324
- tcp_wait
 - host.h, 305
 - tcp.c, 324
- tcp_write
 - host.h, 305
 - tcp.c, 325
- tctreeclear_d
 - symbols.h, 2062
- tctreekeys_d
 - symbols.h, 2062
- tctreevals_d
 - symbols.h, 2062
- tcversion_d
 - symbols.h, 2062
- teacher.c
 - teacher_add_cookie, 2256
 - teacher_add_error, 2256
 - teacher_print_form, 2257
 - teacher_print_message, 2257
 - teacher_process, 2257
- teacher.h
 - teacher_add_cookie, 2259
 - teacher_add_error, 2260
 - teacher_data_delete, 2260
 - teacher_data_fetch, 2260
 - teacher_data_free, 2261
 - teacher_data_get, 2261
 - teacher_data_save, 2261
 - teacher_print_form, 2262
 - teacher_print_message, 2262
 - teacher_process, 2262
- teacher_add_cookie
 - teacher.c, 2256
 - teacher.h, 2259
- teacher_add_error
 - teacher.c, 2256
 - teacher.h, 2260
- teacher_data_delete
 - teacher.h, 2260
 - web/teacher/datatier.c, 719
- teacher_data_fetch
 - teacher.h, 2260
 - web/teacher/datatier.c, 719
- teacher_data_free
 - teacher.h, 2261
 - web/teacher/datatier.c, 720
- teacher_data_get
 - teacher.h, 2261
 - web/teacher/datatier.c, 720
- teacher_data_save
 - teacher.h, 2261
 - web/teacher/datatier.c, 720

- teacher_data_t, 157
 - completed, 157
 - disposition, 157
 - keynum, 157
 - password, 157
 - signature, 157
 - signum, 157
 - username, 158
 - usernum, 158
- teacher_print_form
 - teacher.c, 2257
 - teacher.h, 2262
- teacher_print_message
 - teacher.c, 2257
 - teacher.h, 2262
- teacher_process
 - teacher.c, 2257
 - teacher.h, 2262
- templates
 - content.c, 2109
 - magma_t, 101
- test-internals.c
 - main, 1686
- test-ticks.h
 - maxticks, 1687
 - timeit, 1687
- test.c
 - batch_no_errors, 1689
 - batch_wrong_message, 1689
 - batch_wrong_pk, 1689
 - batch_wrong_sig, 1689
 - batch_point_buffer, 1689
 - batch_test, 1689
 - batch_test_t, 1689
 - curved25519_expected, 1689
 - dataset, 1689
 - main, 1689
 - test_batch_count, 1688
 - test_batch_rounds, 1688
 - test_data, 1689
- test_batch_count
 - test.c, 1688
- test_batch_rounds
 - test.c, 1688
- test_cond
 - testhelp.h, 1707
- test_data
 - test.c, 1689
- test_data_t, 159
 - m, 159
 - pk, 159
 - sig, 159
 - sk, 159
- test_report
 - testhelp.h, 1707
- testhelp.h
 - __failed_tests, 1707
 - __test_num, 1707
 - test_cond, 1707
 - test_report, 1707
- text
 - mail_cache_t, 103
 - mail_message_t, 104
 - smtp_message_t, 147
- thread.h
 - mutex_destroy, 600
 - mutex_init, 600
 - mutex_lock, 601
 - mutex_unlock, 601
 - rwlock_attr_destroy, 602
 - rwlock_attr_getkind, 602
 - rwlock_attr_init, 603
 - rwlock_attr_setkind, 603
 - rwlock_destroy, 603
 - rwlock_init, 604
 - rwlock_lock_read, 604
 - rwlock_lock_write, 604
 - rwlock_unlock, 605
 - thread_alloc, 605
 - thread_cancel, 606
 - thread_cancel_disable, 606
 - thread_cancel_enable, 606
 - thread_get_thread_id, 606
 - thread_join, 606
 - thread_launch, 607
 - thread_result, 607
 - thread_signal, 608
 - tkey_get, 608
 - tkey_init, 608
 - tkey_set, 608
- thread_alloc
 - core/thread/thread.c, 594
 - thread.h, 605
- thread_cancel
 - core/thread/thread.c, 594
 - thread.h, 606
- thread_cancel_disable
 - core/thread/thread.c, 595
 - thread.h, 606
- thread_cancel_enable
 - core/thread/thread.c, 595
 - thread.h, 606
- thread_get_thread_id
 - core/thread/thread.c, 595
 - thread.h, 606
- thread_join
 - core/thread/thread.c, 595
 - thread.h, 606
- thread_launch
 - core/thread/thread.c, 596
 - thread.h, 607
- thread_result
 - core/thread/thread.c, 596
 - thread.h, 607

- thread_signal
 - core/thread/thread.c, 596
 - thread.h, 608
- thread_stack_size
 - magma_t, 101
- thread_start
 - context.h, 769
 - engine/context/thread.c, 598
- thread_stop
 - context.h, 769
 - engine/context/thread.c, 598
- threadBuffer
 - global.c, 728
 - magma.h, 824
- time
 - magma_t, 101
- time.c
 - time_datestamp, 473
 - time_print_gmt, 473
 - time_print_local, 473
 - time_till_midnight, 474
- time_datestamp
 - core/parsers/parsers.h, 450
 - time.c, 473
- time_print_gmt
 - core/parsers/parsers.h, 451
 - time.c, 473
- time_print_local
 - core/parsers/parsers.h, 451
 - time.c, 473
- time_till_midnight
 - core/parsers/parsers.h, 451
 - time.c, 474
- timeit
 - test-ticks.h, 1687
- timeout
 - magma_t, 101
 - server_t, 132
- timestamp
 - cached_object, 33
- tkey_get
 - core/thread/keys.c, 583
 - thread.h, 608
- tkey_init
 - core/thread/keys.c, 583
 - thread.h, 608
- tkey_set
 - core/thread/keys.c, 583
 - thread.h, 608
- TLS
 - cryptography/cryptography.h, 1349
- tls
 - server_t, 132
 - smtp_outbound_prefs_t, 150
- tls.c
 - tls_bits, 1437
 - tls_cipher, 1437
 - tls_client_alloc, 1437
 - tls_continue, 1437
 - tls_error, 1438
 - tls_free, 1438
 - tls_print, 1438
 - tls_read, 1438
 - tls_server_alloc, 1439
 - tls_server_create, 1439
 - tls_server_destroy, 1440
 - tls_status, 1440
 - tls_suite, 1440
 - tls_version, 1440
 - tls_write, 1441
- TLS_PORT
 - engine/config/servers/servers.h, 753
- tls_bits
 - cryptography/cryptography.h, 1375
 - tls.c, 1437
- tls_cipher
 - cryptography/cryptography.h, 1375
 - tls.c, 1437
- tls_client_alloc
 - cryptography/cryptography.h, 1375
 - tls.c, 1437
- tls_continue
 - cryptography/cryptography.h, 1376
 - tls.c, 1437
- tls_ctx
 - server_config_t, 128
- tls_error
 - cryptography/cryptography.h, 1376
 - tls.c, 1438
- tls_free
 - cryptography/cryptography.h, 1376
 - tls.c, 1438
- tls_port
 - server_config_t, 128
- tls_print
 - cryptography/cryptography.h, 1376
 - tls.c, 1438
- tls_read
 - cryptography/cryptography.h, 1377
 - tls.c, 1438
- tls_redirect
 - magma_t, 101
- tls_sd
 - server_config_t, 128
- tls_server_alloc
 - cryptography/cryptography.h, 1377
 - tls.c, 1439
- tls_server_create
 - cryptography/cryptography.h, 1377
 - tls.c, 1439
- tls_server_destroy
 - cryptography/cryptography.h, 1378
 - tls.c, 1440
- tls_status

- cryptography/cryptography.h, 1378
 - tls.c, 1440
- tls_suite
 - cryptography/cryptography.h, 1378
 - tls.c, 1440
- tls_version
 - cryptography/cryptography.h, 1379
 - tls.c, 1440
- tls_write
 - cryptography/cryptography.h, 1379
 - tls.c, 1441
- tlssig
 - dime_record_t, 39
- TLSv1_server_method_d
 - symbols.c, 2008
 - symbols.h, 2062
- to
 - mail_message_t, 105
 - smtp_message_t, 148
- tok_get_bl
 - core/parsers/parsers.h, 451
 - token.c, 478
- tok_get_count_bl
 - core/parsers/parsers.h, 452
 - token.c, 478
- tok_get_count_st
 - core/parsers/parsers.h, 452
 - token.c, 479
- tok_get_ns
 - core/parsers/parsers.h, 453
 - token.c, 479
- tok_get_pl
 - core/parsers/parsers.h, 453
 - token.c, 479
- tok_get_st
 - core/parsers/parsers.h, 453
 - token.c, 480
- tok_pop
 - core/parsers/parsers.h, 454
 - token.c, 480
- tok_pop_init_bl
 - core/parsers/parsers.h, 454
 - token.c, 481
- tok_pop_init_st
 - core/parsers/parsers.h, 454
 - token.c, 481
- tok_state_t, 160
 - block, 160
 - length, 160
 - position, 160
 - remaining, 160
 - token, 160
- token
 - auth_legacy_t, 22
 - auth_t, 26
 - tok_state_t, 160
- token.c
 - pl_get_embraced, 476
 - pl_shrink_before_characters, 476
 - pl_skip_characters, 476
 - pl_skip_to_characters, 476
 - pl_update_start, 477
 - str_tok_get_bl, 477
 - str_tok_get_count_bl, 477
 - tok_get_bl, 478
 - tok_get_count_bl, 478
 - tok_get_count_st, 479
 - tok_get_ns, 479
 - tok_get_pl, 479
 - tok_get_st, 480
 - tok_pop, 480
 - tok_pop_init_bl, 481
 - tok_pop_init_st, 481
- tokens
 - auth_stacie_t, 23
 - auth_t, 26
 - nvp_t, 115
- tokens.c
 - stacie_derive_token, 1956
- tokyo.c
 - lib_load_tokyo, 1971
 - lib_version_tokyo, 1971
- tracing
 - dmime_message_t, 47
- TRACING_HEADER_SIZE
 - dmessage/common.h, 1570
- TRACING_LENGTH_SIZE
 - crypto.h, 1582
- tran_commands
 - transaction.c, 1493
- tran_commit
 - database.h, 1464
 - transaction.c, 1492
- tran_rollback
 - database.h, 1464
 - transaction.c, 1492
- tran_start
 - database.h, 1465
 - transaction.c, 1493
- transaction.c
 - command, 1493
 - length, 1493
 - tran_commands, 1493
 - tran_commit, 1492
 - tran_rollback, 1492
 - tran_start, 1493
- transmit.c
 - smtp_bounce, 2156
 - smtp_forward_message, 2156
 - smtp_relay_message, 2156
 - smtp_reply, 2157
 - smtp_send_message, 2157
- transposition.h
 - PRIME_FIXED_SIZE, 1937

- PRIME_MAX_1_BYTE, [1937](#)
- PRIME_MAX_2_BYTE, [1937](#)
- PRIME_MAX_3_BYTE, [1937](#)
- PRIME_MAX_4_BYTE, [1937](#)
- tree.c
 - __attribute__, [1972](#)
 - tree_alloc, [1972](#)
 - tree_cmp, [1973](#)
 - tree_count, [1973](#)
 - tree_cursor_alloc, [1973](#)
 - tree_cursor_free, [1973](#)
 - tree_cursor_key_active, [1974](#)
 - tree_cursor_key_next, [1974](#)
 - tree_cursor_next, [1974](#)
 - tree_cursor_reset, [1974](#)
 - tree_cursor_t, [1976](#)
 - tree_cursor_value_active, [1974](#)
 - tree_cursor_value_next, [1974](#)
 - tree_delete, [1974](#)
 - tree_find, [1974](#)
 - tree_free, [1975](#)
 - tree_insert, [1975](#)
 - tree_truncate, [1975](#)
- tree_alloc
 - storage.h, [1964](#)
 - tree.c, [1972](#)
- tree_cmp
 - tree.c, [1973](#)
- tree_count
 - storage.h, [1964](#)
 - tree.c, [1973](#)
- tree_cursor_alloc
 - tree.c, [1973](#)
- tree_cursor_free
 - tree.c, [1973](#)
- tree_cursor_key_active
 - tree.c, [1974](#)
- tree_cursor_key_next
 - tree.c, [1974](#)
- tree_cursor_next
 - tree.c, [1974](#)
- tree_cursor_reset
 - tree.c, [1974](#)
- tree_cursor_t
 - tree.c, [1976](#)
- tree_cursor_value_active
 - tree.c, [1974](#)
- tree_cursor_value_next
 - tree.c, [1974](#)
- tree_delete
 - tree.c, [1974](#)
- tree_find
 - tree.c, [1974](#)
- tree_free
 - tree.c, [1975](#)
- tree_insert
 - tree.c, [1975](#)
- tree_truncate
 - tree.c, [1975](#)
- trim.c
 - pl_trim, [482](#)
 - pl_trim_end, [482](#)
 - pl_trim_start, [482](#)
 - st_trim, [482](#)
- ttl
 - cached_object, [33](#)
- tuner
 - tank.c, [1970](#)
- type
 - core.h, [230](#)
 - dmtip_argument_t, [51](#)
 - dmtip_command_t, [53](#)
 - http_content_t, [75](#)
 - mail_mime_t, [107](#)
 - multi_t, [112](#)
 - object_chunk, [117](#)
 - server_t, [132](#)
 - signet_t, [138](#)
 - smtp_inbound_filter_t, [140](#)
 - type.c, [610](#)
- type.c
 - type, [610](#)
- type_embed
 - mysql.c, [1471](#)
- type_serv
 - mysql.c, [1471](#)
- u
 - packedelem32_t, [118](#)
 - packedelem64_t, [119](#)
 - packedelem8_t, [120](#)
- u16
 - multi_t, [112](#)
- u32
 - multi_t, [113](#)
- u64
 - multi_t, [113](#)
- u8
 - multi_t, [113](#)
- uchr_t
 - core.h, [229](#)
- uid
 - imap_fetch_dataitems_t, [79](#)
- uidnext
 - imap_folder_status_t, [82](#)
- uint16_clamp
 - clamp.c, [401](#)
 - numbers.h, [432](#)
- uint16_conv_bl
 - numbers.c, [414](#)
 - numbers.h, [433](#)
- uint16_conv_ns
 - numbers.c, [415](#)
 - numbers.h, [433](#)

- uint16_conv_st
 - numbers.c, [415](#)
 - numbers.h, [433](#)
- uint16_digits
 - digits.c, [405](#)
 - numbers.h, [434](#)
- uint16_get_no
 - numbers.c, [416](#)
 - numbers.h, [434](#)
- uint16_put_no
 - numbers.c, [416](#)
 - numbers.h, [434](#)
- uint24_get_no
 - numbers.c, [416](#)
 - numbers.h, [435](#)
- UINT24_MAX
 - core.h, [228](#)
- UINT24_MIN
 - core.h, [228](#)
- uint24_put_no
 - numbers.c, [416](#)
 - numbers.h, [435](#)
- uint24_t
 - core.h, [229](#)
- uint32_clamp
 - clamp.c, [402](#)
 - numbers.h, [435](#)
- uint32_conv_bl
 - numbers.c, [417](#)
 - numbers.h, [435](#)
- uint32_conv_ns
 - numbers.c, [417](#)
 - numbers.h, [436](#)
- uint32_conv_st
 - numbers.c, [417](#)
 - numbers.h, [436](#)
- uint32_digits
 - digits.c, [405](#)
 - numbers.h, [437](#)
- uint32_get_no
 - numbers.c, [418](#)
 - numbers.h, [437](#)
- uint32_put_no
 - numbers.c, [418](#)
 - numbers.h, [437](#)
- uint64_clamp
 - clamp.c, [402](#)
 - numbers.h, [437](#)
- uint64_conv_bl
 - numbers.c, [418](#)
 - numbers.h, [438](#)
- uint64_conv_ns
 - numbers.c, [419](#)
 - numbers.h, [438](#)
- uint64_conv_pl
 - numbers.c, [419](#)
 - numbers.h, [438](#)
- uint64_conv_st
 - numbers.c, [420](#)
 - numbers.h, [439](#)
- uint64_digits
 - digits.c, [405](#)
 - numbers.h, [439](#)
- uint8_clamp
 - clamp.c, [402](#)
 - numbers.h, [439](#)
- uint8_conv_bl
 - numbers.c, [420](#)
 - numbers.h, [440](#)
- uint8_conv_ns
 - numbers.c, [420](#)
 - numbers.h, [440](#)
- uint8_conv_st
 - numbers.c, [421](#)
 - numbers.h, [440](#)
- uint8_digits
 - digits.c, [406](#)
 - numbers.h, [441](#)
- uint_t
 - core.h, [229](#)
- UNALLOCATED
 - checkers.h, [1262](#)
- uncompress_d
 - symbols.h, [2062](#)
- UNICODE
 - signet/common.h, [1576](#)
- unique
 - dmime_chunk_key_t, [42](#)
 - signet_field_key_t, [135](#)
- unload
 - magma_t, [101](#)
- unpack.c
 - prime_unpack, [1936](#)
 - prime_unpack_fields, [1936](#)
 - prime_unpack_validate, [1936](#)
- unseen
 - imap_folder_status_t, [82](#)
- UNSIGNED_MAX_1_BYTE
 - signet/common.h, [1575](#)
- UNSIGNED_MAX_2_BYTE
 - signet/common.h, [1575](#)
- UNSIGNED_MAX_3_BYTE
 - signet/common.h, [1575](#)
- unused
 - encrypt_keypair_t, [63](#)
- UPDATE_ALERTS_ACKNOWLEDGE
 - queries.h, [2079](#)
- UPDATE_CONTACT
 - queries.h, [2079](#)
- UPDATE_CONTACT_STAMP
 - queries.h, [2080](#)
- UPDATE_FOLDER
 - queries.h, [2080](#)
- UPDATE_LOG_IMAP

- queries.h, 2080
- UPDATE_LOG_POP
 - queries.h, 2080
- UPDATE_LOG_RECEIVED
 - queries.h, 2080
- UPDATE_LOG_SENT
 - queries.h, 2080
- UPDATE_LOG_WEB
 - queries.h, 2080
- UPDATE_MESSAGE_FLAGS_ADD
 - queries.h, 2080
- UPDATE_MESSAGE_FLAGS_REMOVE
 - queries.h, 2080
- UPDATE_MESSAGE_FLAGS_REPLACE
 - queries.h, 2080
- UPDATE_MESSAGE_FOLDER
 - queries.h, 2081
- UPDATE_MESSAGE_VISIBILITY
 - queries.h, 2081
- UPDATE_SIGNATURE_FLAGS_ADD
 - queries.h, 2081
- UPDATE_SIGNATURE_FLAGS_REMOVE
 - queries.h, 2081
- UPDATE_USER_QUOTA_ADD
 - queries.h, 2081
- UPDATE_USER_QUOTA_SUBTRACT
 - queries.h, 2081
- UPDATE_USER_STORAGE_KEYS
 - queries.h, 2081
- updaters.c
 - meta_update_aliases, 1220
 - meta_update_contacts, 1220
 - meta_update_folders, 1221
 - meta_update_keys, 1221
 - meta_update_message_folders, 1222
 - meta_update_realms, 1222
 - meta_update_user, 1222
- upper_chr
 - case.c, 392
 - core/parsers/parsers.h, 454
- upper_st
 - case.c, 393
 - core/parsers/parsers.h, 455
- UPSERT_CONTACT_DETAIL
 - queries.h, 2081
- UPSERT_USER_CONFIG
 - queries.h, 2081
- url.c
 - url_decode, 255
 - url_encode, 255
 - url_valid_chr, 255
 - url_valid_st, 256
- url_decode
 - encodings.h, 246
 - url.c, 255
- url_encode
 - encodings.h, 247
- url.c, 255
- URL_MAX_LENGTH
 - encodings.h, 239
- url_valid_chr
 - encodings.h, 247
 - url.c, 255
- url_valid_st
 - encodings.h, 247
 - url.c, 256
- user
 - __attribute__, 19
 - magma_t, 102
- USER_ALTERNATE_ENCRYPTION_KEY
 - prime.h, 1906
- USER_CONF_STATUS_CRITICAL
 - objects/config/config.h, 677
- USER_CONF_STATUS_NONE
 - objects/config/config.h, 677
- USER_CRYPTO_SIGNATURE
 - prime.h, 1906
- USER_CUSTODY_SIGNATURE
 - prime.h, 1906
- USER_ENCRYPTION_KEY
 - prime.h, 1906
- USER_FULL_SIGNATURE
 - prime.h, 1906
- USER_IDENTIFIABLE_SIGNATURE
 - prime.h, 1906
- USER_IDENTIFIER
 - prime.h, 1906
- USER_SELF_SIGNATURE
 - prime.h, 1906
- USER_SIGNING_KEY
 - prime.h, 1906
- user_config_alloc
 - objects/config/config.c, 1029
 - objects/config/config.h, 678
- user_config_create
 - objects/config/config.c, 1029
 - objects/config/config.h, 678
- user_config_delete
 - objects/config/config.h, 678
 - objects/config/datatier.c, 685
- user_config_edit
 - objects/config/config.c, 1030
 - objects/config/config.h, 679
- user_config_entry_alloc
 - objects/config/config.c, 1030
 - objects/config/config.h, 679
- user_config_entry_free
 - objects/config/config.c, 1030
 - objects/config/config.h, 679
- user_config_entry_t
 - objects/config/config.h, 681
- user_config_fetch
 - objects/config/config.h, 680
 - objects/config/datatier.c, 685

- user_config_free
 - objects/config/config.c, 1031
 - objects/config/config.h, 680
- user_config_t
 - objects/config/config.h, 681
- user_config_update
 - objects/config/config.c, 1031
 - objects/config/config.h, 680
- user_config_upsert
 - objects/config/config.h, 681
 - objects/config/datatier.c, 685
- user_encrypted_key_get
 - keys/users.c, 1867
 - providers/prime/keys/keys.h, 670
- user_encrypted_key_set
 - keys/users.c, 1867
 - providers/prime/keys/keys.h, 670
- user_key_alloc
 - keys/users.c, 1867
 - providers/prime/keys/keys.h, 670
- user_key_free
 - keys/users.c, 1867
 - providers/prime/keys/keys.h, 671
- user_key_generate
 - keys/users.c, 1867
 - providers/prime/keys/keys.h, 671
- user_key_get
 - keys/users.c, 1868
 - providers/prime/keys/keys.h, 671
- user_key_length
 - keys/users.c, 1868
 - providers/prime/keys/keys.h, 671
- user_key_set
 - keys/users.c, 1868
 - providers/prime/keys/keys.h, 671
- user_lock
 - locks.c, 1095
 - objects.h, 1228
- user_request_generate
 - requests.c, 1913
 - signets.h, 1917
- user_request_get
 - requests.c, 1913
 - signets.h, 1917
- user_request_length
 - requests.c, 1913
 - signets.h, 1917
- user_request_rotation
 - requests.c, 1913
 - signets.h, 1917
- user_request_set
 - requests.c, 1914
 - signets.h, 1917
- user_request_sign
 - requests.c, 1914
 - signets.h, 1917
- user_request_verify_chain_of_custody
 - requests.c, 1914
 - signets.h, 1918
- user_request_verify_self
 - requests.c, 1914
 - signets.h, 1918
- user_signet_alloc
 - signets.h, 1918
 - signets/users.c, 1869
- user_signet_fingerprint
 - signets.h, 1918
 - signets/users.c, 1869
- user_signet_free
 - signets.h, 1918
 - signets/users.c, 1869
- user_signet_get
 - signets.h, 1918
 - signets/users.c, 1869
- user_signet_length
 - signets.h, 1919
 - signets/users.c, 1869
- user_signet_set
 - signets.h, 1919
 - signets/users.c, 1870
- user_signet_verify_chain_of_custody
 - signets.h, 1919
 - signets/users.c, 1870
- user_signet_verify_org
 - signets.h, 1919
 - signets/users.c, 1870
- user_signet_verify_self
 - signets.h, 1919
 - signets/users.c, 1870
- user_unlock
 - locks.c, 1095
 - objects.h, 1228
- username
 - __attribute__, 19
 - auth_t, 27
 - register_session_t, 123
 - teacher_data_t, 158
- username.c
 - auth_sanitise_address, 1028
 - auth_sanitise_username, 1028
- usenum
 - __attribute__, 20
 - auth_t, 27
 - register_session_t, 124
 - smtp_inbound_prefs_t, 146
 - smtp_outbound_prefs_t, 150
 - teacher_data_t, 158
- utf8.c
 - lib_load_utf8proc, 1840
 - lib_version_utf8proc, 1840
 - utf8_error_string, 1840
 - utf8_length_st, 1841
 - utf8_valid_st, 1841
- utf8_error_string

- providers/parsers/parsers.h, 459
- utf8.c, 1840
- utf8_length_st
 - providers/parsers/parsers.h, 460
 - utf8.c, 1841
- utf8_valid_st
 - providers/parsers/parsers.h, 460
 - utf8.c, 1841
- utf8proc_category_d
 - symbols.h, 2062
- utf8proc_category_string_d
 - symbols.h, 2062
- utf8proc_errmsg_d
 - symbols.h, 2062
- utf8proc_get_property_d
 - symbols.h, 2062
- utf8proc_iterate_d
 - symbols.h, 2063
- utf8proc_version_d
 - symbols.h, 2063
- v
 - packedelem32_t, 118
 - packedelem64_t, 119
 - packedelem8_t, 120
- va_narg
 - strings.h, 543
- val
 - multi_t, 113
 - statistics_vp_t, 155
- valid
 - generated_data_t, 74
- validate.c
 - st_valid_append, 579
 - st_valid_avail, 579
 - st_valid_destination, 580
 - st_valid_free, 580
 - st_valid_joined, 580
 - st_valid_opts, 581
 - st_valid_placer, 581
 - st_valid_tracked, 581
- validate_dime_record
 - mrec_pub.c, 1768
- validated
 - dime_record_t, 39
 - dnskey, 57
 - ds, 60
- value
 - http_data_t, 76
 - imap_fetch_response_t, 81
- values
 - engine/status/statistics.c, 805
 - mappings_t, 108
- verification
 - auth_stacie_t, 23
 - auth_t, 27
- verify_dx_certificate
 - dmtp_pub.c, 1729
- verify_ec_sha_signature
 - crypto_pub.c, 1513
- verify_ec_signature
 - crypto_pub.c, 1513
- version
 - dime_record_t, 39
- violations
 - server_t, 132
- virus
 - magma_t, 102
 - smtp_inbound_prefs_t, 146
 - smtp_session_t, 154
- virus_check
 - checkers.h, 1268
 - clamav.c, 1272
- virus_engine
 - clamav.c, 1274
- virus_engine_create
 - checkers.h, 1268
 - clamav.c, 1272
- virus_engine_destroy
 - checkers.h, 1269
 - clamav.c, 1272
- virus_engine_refresh
 - checkers.h, 1269
 - clamav.c, 1273
- virus_lock
 - clamav.c, 1274
- virus_sigs
 - clamav.c, 1274
- virus_sigs_loaded
 - checkers.h, 1269
 - clamav.c, 1273
- virus_sigs_total
 - checkers.h, 1269
 - clamav.c, 1273
- virus_spool
 - clamav.c, 1274
- virus_start
 - checkers.h, 1270
 - clamav.c, 1273
- virus_stat
 - clamav.c, 1274
- virus_stop
 - checkers.h, 1270
 - clamav.c, 1274
- virusaction
 - smtp_inbound_prefs_t, 146
- warehouse.c
 - warehouse_start, 1250
 - warehouse_stop, 1250
 - warehouse_update, 1250
- warehouse.h
 - __attribute__, 1253
 - domain_alloc, 1253

- domain_dkim, 1253
- domain_mailboxes, 1254
- domain_restricted, 1254
- domain_spf, 1254
- domain_start, 1254
- domain_stop, 1255
- domain_t, 1258
- domain_wildcard, 1255
- pattern_check, 1255
- pattern_start, 1255
- pattern_stop, 1256
- pattern_update, 1256
- warehouse_fetch_domains, 1256
- warehouse_fetch_patterns, 1256
- warehouse_start, 1257
- warehouse_stop, 1257
- warehouse_update, 1257
- warehouse_fetch_domains
 - objects/warehouse/datatier.c, 707
 - warehouse.h, 1256
- warehouse_fetch_patterns
 - objects/warehouse/datatier.c, 707
 - warehouse.h, 1256
- warehouse_start
 - warehouse.c, 1250
 - warehouse.h, 1257
- warehouse_stop
 - warehouse.c, 1250
 - warehouse.h, 1257
- warehouse_update
 - warehouse.c, 1250
 - warehouse.h, 1257
- web
 - magma_t, 102
- web/portal/config.c
 - portal_config_collection, 1032
 - portal_config_entry, 1032
 - portal_config_entry_flags, 1032
- web/portal/contacts.c
 - portal_contact_detail_flags, 1043
 - portal_contact_details, 1043
- web/portal/flags.c
 - portal_message_flags_array, 2122
 - portal_message_tags_array, 2122
 - portal_parse_flags, 2122
- web/portal/folders.c
 - portal_folder_contacts_add, 1070
 - portal_folder_contacts_remove, 1070
 - portal_folder_mail_add, 1071
 - portal_folder_mail_remove, 1071
- web/portal/messages.c
 - portal_message_attachments, 1091
 - portal_message_body, 1091
 - portal_message_header, 1091
 - portal_message_info, 1092
 - portal_message_meta, 1092
 - portal_message_security, 1092
 - portal_message_server, 1092
 - portal_message_source, 1093
- web/portal/parse.c
 - portal_parse_action, 2098
 - portal_parse_context, 2098
 - portal_parse_json_str_array, 2098
 - portal_parse_sections, 2099
- web/register/datatier.c
 - register_data_check_username, 715
 - register_data_fetch_blocklist, 715
 - register_data_insert_user, 715
- web/register/sessions.c
 - register_session_cache, 1242
 - register_session_free, 1242
 - register_session_generate, 1243
 - register_session_get, 1243
- web/statistics/datatier.c
 - portal_statistics_mutex, 718
 - PORTAL_STATISTICS_TIMEOUT, 717
 - portal_stats, 718
 - statistics_init, 717
 - statistics_last_updated, 718
 - statistics_refresh, 717
- web/statistics/statistics.c
 - portal_stats, 806
 - statistics_process, 806
- web/teacher/datatier.c
 - teacher_data_delete, 719
 - teacher_data_fetch, 719
 - teacher_data_free, 720
 - teacher_data_get, 720
 - teacher_data_save, 720
- weight
 - allocations.h, 1259
 - cache_t, 30
- worker_threads
 - magma_t, 102
- workers
 - queue.c, 786
- wrap_line_length
 - magma_t, 102
- write.c
 - client_print, 1009
 - client_write, 1009
 - con_print, 1010
 - con_write_bl, 1010
 - con_write_ns, 1011
 - con_write_pl, 1011
 - con_write_st, 1012
- write51
 - curve25519-donna-64bit.h, 1640
- write51full
 - curve25519-donna-64bit.h, 1640
- X
 - ge_p1p1, 70
 - ge_p2, 71

- ge_p3, [72](#)
- x
 - ge25519_plp1_t, [66](#)
 - ge25519_t, [68](#)
 - X509_check_host_d
 - symbols.h, [2063](#)
 - X509_check_issued_d
 - symbols.c, [2008](#)
 - symbols.h, [2063](#)
 - X509_email_free_d
 - symbols.c, [2008](#)
 - symbols.h, [2063](#)
 - X509_get1_ocsp_d
 - symbols.c, [2008](#)
 - symbols.h, [2063](#)
 - X509_get_ext_count_d
 - symbols.c, [2008](#)
 - symbols.h, [2063](#)
 - X509_get_ext_d
 - symbols.c, [2009](#)
 - symbols.h, [2063](#)
 - X509_get_subject_name_d
 - symbols.c, [2009](#)
 - symbols.h, [2063](#)
 - X509_LOOKUP_file_d
 - symbols.c, [2009](#)
 - symbols.h, [2063](#)
 - X509_NAME_ENTRY_get_data_d
 - symbols.h, [2063](#)
 - X509_NAME_get_entry_d
 - symbols.h, [2063](#)
 - X509_NAME_get_index_by_NID_d
 - symbols.h, [2063](#)
 - X509_NAME_get_text_by_NID_d
 - symbols.h, [2064](#)
 - X509_NAME_online_d
 - symbols.c, [2009](#)
 - symbols.h, [2064](#)
 - X509_STORE_add_lookup_d
 - symbols.h, [2064](#)
 - X509_STORE_CTX_free_d
 - symbols.c, [2009](#)
 - symbols.h, [2064](#)
 - X509_STORE_CTX_get_current_cert_d
 - symbols.h, [2064](#)
 - X509_STORE_CTX_get_error_d
 - symbols.c, [2009](#)
 - symbols.h, [2064](#)
 - X509_STORE_CTX_get_error_depth_d
 - symbols.c, [2009](#)
 - symbols.h, [2064](#)
 - X509_STORE_CTX_init_d
 - symbols.h, [2064](#)
 - X509_STORE_CTX_new_d
 - symbols.c, [2009](#)
 - symbols.h, [2064](#)
 - X509_STORE_CTX_set_chain_d
 - symbols.h, [2064](#)
 - X509_STORE_free_d
 - symbols.c, [2009](#)
 - symbols.h, [2064](#)
 - X509_STORE_load_locations_d
 - symbols.h, [2064](#)
 - X509_STORE_new_d
 - symbols.c, [2009](#)
 - symbols.h, [2065](#)
 - X509_STORE_set_flags_d
 - symbols.h, [2065](#)
 - X509_verify_cert_d
 - symbols.c, [2009](#)
 - symbols.h, [2065](#)
 - X509_verify_cert_error_string_d
 - symbols.c, [2009](#)
 - symbols.h, [2065](#)
- xaddy
 - ge25519_niels_t, [65](#)
 - ge25519_pniels_t, [67](#)
- xml.c
 - lib_load_xml, [1844](#)
 - lib_version_xml, [1844](#)
 - log_mutex, [1854](#)
 - xml_create_doc, [1844](#)
 - xml_create_parser_ctx, [1845](#)
 - xml_create_xpath_ctx, [1845](#)
 - xml_dump_doc, [1845](#)
 - xml_encode, [1846](#)
 - xml_error, [1846](#)
 - xml_free_doc, [1846](#)
 - xml_free_parser_ctx, [1846](#)
 - xml_free_xpath_ctx, [1847](#)
 - xml_free_xpath_obj, [1847](#)
 - xml_get_xpath_int16, [1847](#)
 - xml_get_xpath_int32, [1848](#)
 - xml_get_xpath_int64, [1848](#)
 - xml_get_xpath_int8, [1848](#)
 - xml_get_xpath_node_count, [1848](#)
 - xml_get_xpath_ns, [1849](#)
 - xml_get_xpath_st, [1849](#)
 - xml_get_xpath_uint16, [1849](#)
 - xml_get_xpath_uint32, [1849](#)
 - xml_get_xpath_uint64, [1850](#)
 - xml_get_xpath_uint8, [1850](#)
 - xml_node_add_sibling, [1850](#)
 - xml_node_free, [1851](#)
 - xml_node_get_content_st, [1851](#)
 - xml_node_new, [1851](#)
 - xml_node_set_content, [1852](#)
 - xml_node_set_property, [1852](#)
 - xml_set_xpath_ns, [1852](#)
 - xml_set_xpath_property, [1852](#)
 - xml_set_xpath_uint64, [1853](#)
 - xml_start, [1853](#)
 - xml_stop, [1853](#)
 - xml_version_string, [1854](#)

- xml_xpath_eval, 1854
- xml_xpath_set_namespace, 1854
- xml_create_doc
 - providers/parsers/parsers.h, 460
 - xml.c, 1844
- xml_create_parser_ctx
 - providers/parsers/parsers.h, 461
 - xml.c, 1845
- xml_create_xpath_ctx
 - providers/parsers/parsers.h, 461
 - xml.c, 1845
- xml_dump_doc
 - providers/parsers/parsers.h, 461
 - xml.c, 1845
- xml_encode
 - providers/parsers/parsers.h, 462
 - xml.c, 1846
- xml_error
 - providers/parsers/parsers.h, 462
 - xml.c, 1846
- xml_free_doc
 - providers/parsers/parsers.h, 462
 - xml.c, 1846
- xml_free_parser_ctx
 - providers/parsers/parsers.h, 462
 - xml.c, 1846
- xml_free_xpath_ctx
 - providers/parsers/parsers.h, 463
 - xml.c, 1847
- xml_free_xpath_obj
 - providers/parsers/parsers.h, 463
 - xml.c, 1847
- xml_get_xpath_int16
 - providers/parsers/parsers.h, 463
 - xml.c, 1847
- xml_get_xpath_int32
 - providers/parsers/parsers.h, 464
 - xml.c, 1848
- xml_get_xpath_int64
 - providers/parsers/parsers.h, 464
 - xml.c, 1848
- xml_get_xpath_int8
 - providers/parsers/parsers.h, 464
 - xml.c, 1848
- xml_get_xpath_node_count
 - providers/parsers/parsers.h, 464
 - xml.c, 1848
- xml_get_xpath_ns
 - providers/parsers/parsers.h, 465
 - xml.c, 1849
- xml_get_xpath_st
 - providers/parsers/parsers.h, 465
 - xml.c, 1849
- xml_get_xpath_uint16
 - providers/parsers/parsers.h, 465
 - xml.c, 1849
- xml_get_xpath_uint32
 - providers/parsers/parsers.h, 465
 - xml.c, 1849
- xml_get_xpath_uint64
 - providers/parsers/parsers.h, 466
 - xml.c, 1850
- xml_get_xpath_uint8
 - providers/parsers/parsers.h, 466
 - xml.c, 1850
- xml_node_add_sibling
 - providers/parsers/parsers.h, 466
 - xml.c, 1850
- xml_node_free
 - providers/parsers/parsers.h, 467
 - xml.c, 1851
- xml_node_get_content_st
 - providers/parsers/parsers.h, 467
 - xml.c, 1851
- xml_node_new
 - providers/parsers/parsers.h, 467
 - xml.c, 1851
- xml_node_set_content
 - providers/parsers/parsers.h, 468
 - xml.c, 1852
- xml_node_set_property
 - providers/parsers/parsers.h, 468
 - xml.c, 1852
- xml_set_xpath_ns
 - providers/parsers/parsers.h, 468
 - xml.c, 1852
- xml_set_xpath_property
 - providers/parsers/parsers.h, 468
 - xml.c, 1852
- xml_set_xpath_uint64
 - providers/parsers/parsers.h, 469
 - xml.c, 1853
- xml_start
 - providers/parsers/parsers.h, 469
 - xml.c, 1853
- xml_stop
 - providers/parsers/parsers.h, 469
 - xml.c, 1853
- xml_version_string
 - xml.c, 1854
- xml_xpath_eval
 - providers/parsers/parsers.h, 470
 - xml.c, 1854
- xml_xpath_set_namespace
 - providers/parsers/parsers.h, 470
 - xml.c, 1854
- xmlAddSibling_d
 - symbols.h, 2065
- xmlBufferContent_d
 - symbols.h, 2065
- xmlBufferCreate_d
 - symbols.h, 2065
- xmlBufferFree_d
 - symbols.h, 2065

- xmlBufferLength_d
 - symbols.h, [2065](#)
- xmlCleanupGlobals_d
 - symbols.h, [2065](#)
- xmlCleanupParser_d
 - symbols.h, [2065](#)
- xmlCtxtReadMemory_d
 - symbols.h, [2065](#)
- xmlDocDumpFormatMemory_d
 - symbols.h, [2066](#)
- xmlEncodeEntitiesReentrant_d
 - symbols.h, [2066](#)
- xmlFreeDoc_d
 - symbols.h, [2066](#)
- xmlFreeNode_d
 - symbols.h, [2066](#)
- xmlFreeParserCtxt_d
 - symbols.h, [2066](#)
- xmlInitParser_d
 - symbols.h, [2066](#)
- xmlMemoryDump_d
 - symbols.h, [2066](#)
- xmlNewNode_d
 - symbols.h, [2066](#)
- xmlNewParserCtxt_d
 - symbols.h, [2066](#)
- xmlNodeBufGetContent_d
 - symbols.h, [2066](#)
- xmlNodeSetContent_d
 - symbols.h, [2066](#)
- xmlParserVersion_d
 - symbols.h, [2066](#)
- xmlSetProp_d
 - symbols.h, [2067](#)
- xmlXPathEvalExpression_d
 - symbols.h, [2067](#)
- xmlXPathFreeContext_d
 - symbols.h, [2067](#)
- xmlXPathFreeObject_d
 - symbols.h, [2067](#)
- xmlXPathNewContext_d
 - symbols.h, [2067](#)
- xmlXPathRegisterNs_d
 - symbols.h, [2067](#)
- xmimi
 - curve25519-donna-sse2.h, [1643](#)
- xpath_ctx
 - http_page_t, [77](#)
- xy2d
 - ge_precomp, [73](#)
- Y
 - ge_p1p1, [70](#)
 - ge_p2, [71](#)
 - ge_p3, [72](#)
- y
 - ge25519_p1p1_t, [66](#)
 - ge25519_t, [68](#)
- YminusX
 - ge_cached, [69](#)
- yminusx
 - ge_precomp, [73](#)
- YplusX
 - ge_cached, [69](#)
- yplusx
 - ge_precomp, [73](#)
- ysubx
 - ge25519_niels_t, [65](#)
 - ge25519_pniels_t, [67](#)
- Z
 - ge_cached, [69](#)
 - ge_p1p1, [70](#)
 - ge_p2, [71](#)
 - ge_p3, [72](#)
- z
 - ge25519_p1p1_t, [66](#)
 - ge25519_pniels_t, [67](#)
 - ge25519_t, [68](#)
- zbase32
 - mappings_t, [108](#)
- zbase32.c
 - zbase32_decode, [257](#)
 - zbase32_encode, [257](#)
- zbase32_decode
 - encodings.h, [247](#)
 - zbase32.c, [257](#)
- zbase32_encode
 - encodings.h, [248](#)
 - zbase32.c, [257](#)
- zlib.c
 - compress_zlib, [1305](#)
 - decompress_zlib, [1305](#)
 - deflate_d, [1306](#)
 - deflateEnd_d, [1306](#)
 - deflateInit2__d, [1306](#)
 - lib_load_zlib, [1306](#)
 - lib_version_zlib, [1306](#)
- zlibVersion_d
 - symbols.h, [2067](#)