ZebOS-XP®
Network Platform

Version 1.4
Extended Performance

Multicast
Developer Guide

December 2015

IP Infusion Inc.
3965 Freedom Circle, Suite 200
Santa Clara, CA 95054
+1 408-400-1900
http://www.ipinfusion.com/

For support, questions, or comments via E-mail, contact:
support@ipinfusion.com

Contents

Preface xiii
Audience xiii
Conventions xiii
Contents xiii
Related Documents xiv
Support xiv
Comments xiv

CHAPTER 1 ZebOS-XP Multicast Protocols and MRIB 15
Multicast Architecture Overview 15
MRIBd 15
Code Structure of MRIB 15
MRIB Services 17
Multicast-Related Modules 17
Multicast Protocol and MRIB Message Exchange 18
IGMP 19

CHAPTER 2 ZebOS-XP and Multicast Forwarder Interaction 21
Multicast Forwarder Programming 21
Multicast Forwarder Events 22

CHAPTER 3 Multiple Registration Protocols 23
Overview 23
Architecture 23
Functionality 23
Features 24
MRP Data Structures 24
API 26
gmrp_set_timer 26
gmrp_disable 27
gmrp_disable_port 28
gmrp_get_per_vlan_statistics_details 29
gmrp_clear_all_statistics 30
gmrp_clear_per_vlan_statistics 30
gmrp_disable_instance 31
gmrp_enable_instance 32

CHAPTER 4 Data Structures 33
Common Data Structures 33
Multicast Data Structures 33
mrib_snmp_api_vif_index 33
mrib_snmp_api_mrt_index 33
igmp_instance 38
mld_instance 40

CHAPTER 5 MRIBv4 Command API 49
API 49
Include File 49
mrib4_api_multicast_routing_set 49
mrib4_api_multicast_routing_unset 50
mrib4_api_rt_limit_thresh_set 51
mrib4_api_rt_limit_thresh_unset 52
mrib4_api_rt_limit_set 53
mrib4_api_vif_ttl_threshold_set 54
mrib4_api_vif_ttl_threshold_unset 55
mrib4_api_clear_mroute_all 56
mrib4_api_clear_mroute_g 57
mrib4_api_clear_mroute_sg 58
mrib4_api_clear_mroute_stats_all 59
mrib4_api_clear_mroute_stats_g 60
mrib4_api_clear_mroute_stats_sg 61
mrib4_api_debug_all_set 62
mrib4_api_debug_all_unset 63
mrib4_api_debug_event_set 64
mrib4_api_debug_event_unset 65
mrib4_api_debug_vif_set 66
mrib4_api_debug_vif_unset 67
mrib4_api_debug_mrt_set 68
mrib4_api_debug_mrt_unset 69
mrib4_api_debug_stats_set 70
mrib4_api_debug_stats_unset 71
mrib4_api_debug_fib_msg_set 72
mrib4_api_debug_fib_msg_unset 73
mrib4_api_debug_register_msg_set 74
mrib4_api_debug_register_msg_unset 75
mrib4_api_debug_nsm_msg_set 76
mrib4_api_debug_nsm_msg_unset 77
mrib4_api_debug_mrib_msg_set 78
mrib4_api_debug_mrib_msg_unset 79
mrib4_api_debug_mtrace_set 80
mrib4_api_debug_mtrace_unset 81
mrib4_api_debug_mtrace_detail_set 82
mrib4_api_debug_mtrace_detail_unset 83

CHAPTER 6 MRIBv6 Command API 85
API 85
Include File 85
mrib6_api_multicast_routing_set 85

CHAPTER 7 Multicast Routing MIB API 119
Tables 119
IP Multicast Route Table 119
IP Multicast Routing Next Hop Table 119

IP Multicast Routing Interface Table 119
Tables Not Supported 119
API 119
Include File. 119
mrib_snmp_api_get_mcast_enable 119
mrib_snmp_api_set_mcast_enable 120
mrib_snmp_api_get_next_vif_ttl 122
mrib_snmp_api_get_vif_ratelimit 123
mrib_snmp_api_get_next_vif_ratelimit 123
mrib_snmp_api_set_vif_ratelimit 124
mrib_snmp_api_get_vif_inmcastoctets 124
mrib_snmp_api_get_next_vif_outmcastoctets. 126
mrib_snmp_api_get_next_vif_inmcastpkts 127
mrib_snmp_api_get_vif_outmcastpkts. 128
mrib_snmp_api_get_next_vif_outmcastpkts 128
mrib_snmp_api_get_vif_discontinuity_time 129
mrib_snmp_api_get_next_vif_discontinuity_time 130
mrib_snmp_api_get_mrt_upstreamneighbor_type 131
mrib_snmp_api_get_next_mrt_upstreamneighbor_type 132
mrib_snmp_api_get_mrt_upstreamneighbor 133
mrib_snmp_api_get_next_mrt_upstreamneighbor 134
mrib_snmp_api_get_mrt_inifindex 135
mrib_snmp_api_get_next_mrt_inifindex 136
mrib_snmp_api_get_mrt_timestamp 137
mrib_snmp_api_get_next_mrt_timestamp. 138
mrib_snmp_api_get_mrt_expirytime 139
mrib_snmp_api_get_next_mrt_expirytime 140
mrib_snmp_api_get_mrt_protocol 141
mrib_snmp_api_get_next_mrt_protocol. 142
mrib_snmp_api_get_mrt_rtprotocol 143
mrib_snmp_api_get_next_mrt_rtprotocol 144
mrib_snmp_api_get_mrt_rtaddr_type 145
mrib_snmp_api_get_next_mrt_rtaddr_type 146
mrib_snmp_api_get_mrt_rtaddr. 147
mrib_snmp_api_get_next_mrt_rtaddr 148
mrib_snmp_api_get_mrt_rtprefix_len 149
mrib_snmp_api_get_next_mrt_rtprefix_len 150
mrib_snmp_api_get_mrt_rttype 151
mrib_snmp_api_get_next_mrt_rttype 152
mrib_snmp_api_get_mrt_octets. 153
mrib_snmp_api_get_next_mrt_octets 154
mrib_snmp_api_get_mrt_pkts 155
mrib_snmp_api_get_next_mrt_pkts. 156
mrib_snmp_api_get_mrt_diffinifoctets 157
mrib_snmp_api_get_next_mrt_diffinifoctets 158
mrib_snmp_api_get_nh_state 159
mrib_snmp_api_get_next_nh_state. 160

mrib_snmp_api_get_nh_uptime . . . 161
mrib_snmp_api_get_next_nh_uptime . . . 162
mrib_snmp_api_get_nh_expirytime . . . 163
mrib_snmp_api_get_next_nh_expirytime . . . 164
mrib_snmp_api_get_nh_closestmemberhops . . . 165
mrib_snmp_api_get_next_nh_closestmemberhops . . . 166
mrib_snmp_api_get_nh_protocol . . . 167
mrib_snmp_api_get_next_nh_protocol . . . 168
mrib_snmp_api_get_nh_octets . . . 169
mrib_snmp_api_get_next_nh_octets . . . 170
mrib_snmp_api_get_nh_pkts . . . 171
mrib_snmp_api_get_next_nh_pkts . . . 172

CHAPTER 8 MRIBv4 IGMP Command API . . . 173
Include File . . . 173
Instance-Level Configuration API . . . 173
mrib4_igmp_api_limit_set . . . 173
mrib4_igmp_api_ssm_map_enable_set . . . 174
mrib4_igmp_api_ssm_map_enable_unset . . . 175
mrib4_igmp_api_ssm_map_static_set . . . 175
mrib4_igmp_api_ssm_map_static_unset . . . 176
mrib4_igmp_api_clear . . . 176
Interface-Level Configuration API . . . 177
mrib4_igmp_api_if_set . . . 177
mrib4_igmp_api_if_unset . . . 178
mrib4_igmp_api_if_access_list_set . . . 179
mrib4_igmp_api_if_lmqc_set . . . 183
mrib4_igmp_api_if_lmqc_unset . . . 184
mrib4_igmp_api_if_lmqi_unset . . . 186
mrib4_igmp_api_if_mroute_pxy_unset . . . 188
mrib4_igmp_api_if_querier_timeout_set . . . 191
mrib4_igmp_api_offlink_if_set . . . 201
mrib4_igmp_api_offlink_if_unset . . . 202
mrib4_igmp_api_if_startup_query_interval_set . . . 203
mrib4_igmp_api_if_startup_query_interval_unset . . . 204
mrib4_igmp_api_if_startup_query_count_set . . . 205
mrib4_igmp_api_if_startup_query_count_unset . . . 206
mrib4_igmp_api_if_ra_set . . . 207
mrib4_igmp_api_if_ra_unset . . . 208
mrib4_igmp_api_if_static_join_group_source_set . . . 209

CHAPTER 9 L2 IGMP Snooping Command API . . . 213
Overview . . . 213
How L2mribd Interacts with Protocols . . . 213
Include File . . . 214
Instance-Level Configuration API . . . 214
igmp_snooping_set . . . 215
igmp_snooping_unset . . . 215

Interface-Level Configuration API 215
igmp_if_snooping_set 216
igmp_if_snooping_unset 216
igmp_if_snoop_fast_leave_unset 218
igmp_if_snoop_mrouter_if_set. 219
igmp_if_snoop_querier_unset 221
igmp_if_snoop_report_suppress_set. 222

CHAPTER 10 MRIBv4 IGMP MIB API 225
Include File 225
Return Values 225
IGMP MIB Table 225
mrib4_igmp_snmp_api_if_querier_get. 225
mrib4_igmp_snmp_api_if_querier_get_next 226
mrib4_igmp_snmp_api_if_query_interval_get 226
mrib4_igmp_snmp_api_if_query_interval_get_next 227
mrib4_igmp_snmp_api_if_status_get 227
mrib4_igmp_snmp_api_if_status_get_next 228
mrib4_igmp_snmp_api_if_wrong_version_queries_get. 229
igmp_if_wrong_version_queries_get_next 229
mrib4_igmp_snmp_api_if_version_get 230
mrib4_igmp_snmp_api_if_version_get_next 231
mrib4_igmp_snmp_api_if_query_response_interval_get 231
mrib4_igmp_snmp_api_if_query_response_interval_get_next 232
igmp_if_querier_uptime_get 232
mrib4_igmp_snmp_api_if_querier_uptime_get_next 233
mrib4_igmp_snmp_api_if_querier_expiry_time_get_next 234
mrib4_igmp_snmp_api_if_mroute_pxy_get 235
mrib4_igmp_snmp_api_if_mroute_pxy_get_next 235
mrib4_igmp_snmp_api_if_robustness_var_get 236
mrib4_igmp_snmp_api_if_robustness_var_get_next 236
mrib4_igmp_snmp_api_if_lmqi_get 237
mrib4_igmp_snmp_api_if_lmqi_get_next 238
mrib4_igmp_snmp_api_if_lmqc_get 238
mrib4_igmp_snmp_api_if_lmqc_get_next 239
mrib4_igmp_snmp_api_if_sqc_get 239
mrib4_igmp_snmp_api_if_sqc_get_next 240
mrib4_igmp_snmp_api_if_sqi_get 240
mrib4_igmp_snmp_api_if_sqi_get_next 241
lmrib4_igmp_snmp_api_if_srclist_host_address_get 242
mrib4_igmp_snmp_api_if_srclist_host_address_get_next 242
mrib4_igmp_snmp_api_if_srclist_expiry_time_get 243
mrib4_igmp_snmp_api_if_srclist_expiry_time_get_next 244
mrib4_igmp_snmp_api_if_joins_get 244
mrib4_igmp_snmp_api_if_joins_get_next 245
mrib4_igmp_snmp_api_if_groups_get. 246
mrib4_igmp_snmp_api_if_groups_get_next 246

IGMP Cache MIB Table . . . 247
mrib4_igmp_snmp_api_if_cache_last_reporter_get . . . 247
mrib4_igmp_snmp_api_if_cache_last_reporter_get_next . . . 247
mrib4_igmp_snmp_api_if_cache_uptime_get . . . 248
mrib4_igmp_snmp_api_if_cache_uptime_get_next . . . 249
mrib4_igmp_snmp_api_if_cache_expiry_time_get . . . 249
mrib4_igmp_snmp_api_if_cache_expiry_time_get_next . . . 250
mrib4_igmp_snmp_api_if_cache_exclmode_exp_timer_get_next . . . 252
mrib4_igmp_snmp_api_if_cache_ver1_host_timer_get . . . 253
mrib4_igmp_snmp_api_if_cache_ver1_host_timer_get_next . . . 254
mrib4_igmp_snmp_api_if_cache_ver2_host_timer_get_next . . . 255
mrib4_igmp_snmp_api_if_cache_src_filter_mode_get . . . 256
mrib4_igmp_snmp_api_if_inv_cache_address_get . . . 257
mrib4_igmp_snmp_api_if_inv_cache_address_get_next . . . 258
IGMP API Error Codes . . . 259
mrib4_igmp_snmp_api_strerror . . . 260

CHAPTER 11 MRIBv6 MLD Command API . . . 261
Include File . . . 261
Global Configuration API . . . 261
mrib6_mld_api_limit_set . . . 261
mrib6_mld_api_limit_unset . . . 261
mrib6_mld_api_ssm_map_enable_set . . . 262
mrib6_mld_api_ssm_map_enable_unset . . . 262
mrib6_mld_api_ssm_map_static_set . . . 263
mrib6_mld_api_ssm_map_static_unset . . . 263
Interface-Level Configuration API . . . 264
mrib6_mld_api_if_set . . . 264
mrib6_mld_api_if_unset . . . 265
mrib6_mld_api_if_access_list_set . . . 265
mrib6_mld_api_if_access_list_unset . . . 266
mrib6_mld_api_if_immediate_leave_set . . . 267
mrib6_mld_api_if_immediate_leave_unset . . . 267
mrib6_mld_api_if_limit_set . . . 268
mrib6_mld_api_if_limit_unset . . . 269
mrib6_mld_api_if_lmqc_set . . . 270
mrib6_mld_api_if_lmqc_unset . . . 270
mrib6_mld_api_if_lmqi_set . . . 271
mrib6_mld_api_if_lmqi_unset . . . 272
mrib6_mld_api_if_mroute_pxy_set . . . 273
mrib6_mld_api_if_mroute_pxy_unset . . . 273
mrib6_mld_api_if_pxy_service_set . . . 274
mrib6_mld_api_if_pxy_service_unset . . . 275
mrib6_mld_api_if_querier_timeout_set . . . 275
mrib6_mld_api_if_querier_timeout_unset . . . 276
mrib6_mld_api_if_query_interval_set . . . 277
mrib6_mld_api_if_query_interval_unset . . . 278

mrib6_mld_api_if_query_response_interval_set 278
mrib6_mld_api_if_query_response_interval_unset 279
mrib6_mld_api_if_robustness_var_set 280
mrib6_mld_api_if_robustness_var_unset 281
mrib6_mld_api_if_version_set 281
mrib6_mld_api_if_version_unset 282
mrib6_mld_api_if_static_group_source_set 283
mrib6_mld_api_if_static_group_source_unset 284
Clear Configuration API 285
mrib6_mld_api_clear 285

CHAPTER 12 L2 MLD Snooping Command API 287
Include File 287
Instance-Level Configuration API 287
mld_snooping_set 287
mld_snooping_unset 287
Interface-Level Configuration API 288
mld_if_snooping_set 288
mld_if_snooping_unset 289
mld_if_snoop_fast_leave_set 289
mld_if_snoop_fast_leave_unset 290
mld_if_snoop_mrouter_if_set 291
mld_if_snoop_mrouter_if_unset 292
mld_if_snoop_querier_set 293
mld_if_snoop_querier_unset 293
mld_if_snoop_report_suppress_set 294
mld_if_snoop_report_suppress_unset 295

CHAPTER 13 MRIBv6 MLD MIB API 297
Include File 297
Return Values 297
API 297
mrib6_mld_snmp_api_if_querier_get 297
mrib6_mld_snmp_api_if_querier_get_next 298
mrib6_mld_snmp_api_if_query_interval_get 298
mrib6_mld_snmp_api_if_status_get 300
mrib6_mld_snmp_api_if_status_get_next 301
mrib6_mld_snmp_api_if_wrong_version_queries_get_next 302
mrib6_mld_snmp_api_if_joins_get 303
mrib6_mld_snmp_api_if_joins_get_next 303
mrib6_mld_snmp_api_if_groups_get 304
mrib6_mld_snmp_api_if_version_get 305
mrib6_mld_snmp_api_if_version_get_next 306
mrib6_mld_snmp_api_if_query_response_interval_get 306
mrib6_mld_snmp_api_if_query_response_interval_get_next 307
mrib6_mld_snmp_api_if_querier_uptime_get 307
mrib6_mld_snmp_api_if_querier_uptime_get_next 308
mrib6_mld_snmp_api_if_querier_expiry_time_get_next 309

mrib6_mld_snmp_api_if_mroute_pxy_get . . . 310
mrib6_mld_snmp_api_if_mroute_pxy_get_next . . . 310
mrib6_mld_snmp_api_if_robustness_var_get . . . 311
mrib6_mld_snmp_api_if_robustness_var_get_next . . . 311
mrib6_mld_snmp_api_if_lmqi_get . . . 312
mrib6_mld_snmp_api_if_lmqc_get . . . 313
mrib6_mld_snmp_api_if_lmqc_get_next . . . 314
mrib6_mld_snmp_api_if_sqc_get . . . 314
mrib6_mld_snmp_api_if_sqc_get_next . . . 315
mrib6_mld_snmp_api_if_sqi_get . . . 315
mrib6_mld_snmp_api_if_sqi_get_next . . . 316
mrib6_mld_snmp_api_if_cache_last_reporter_get . . . 317
mrib6_mld_snmp_api_if_cache_last_reporter_get_next . . . 317
mrib6_mld_snmp_api_if_cache_uptime_get . . . 318
mrib6_mld_snmp_api_if_cache_uptime_get_next . . . 319
mrib6_mld_snmp_api_if_cache_expiry_time_get . . . 319
mrib6_mld_snmp_api_if_cache_expiry_time_get_next . . . 320
mrib6_mld_snmp_api_if_cache_exclmode_exp_timer_get . . . 321
mrib6_mld_snmp_api_if_cache_exclmode_exp_timer_get_next . . . 321
mrib6_mld_snmp_api_if_cache_ver1_host_timer_get . . . 322
mrib6_mld_snmp_api_if_cache_ver1_host_timer_get_next . . . 323
mrib6_mld_snmp_api_if_cache_ver2_host_timer_get . . . 323
mrib6_mld_snmp_api_if_cache_ver2_host_timer_get_next . . . 324
mrib6_mld_snmp_api_if_cache_src_filter_mode_get_next . . . 325
mrib6_mld_snmp_api_if_inv_cache_address_get . . . 326
mrib6_mld_snmp_api_if_srclist_host_address_get . . . 327
mrib6_mld_snmp_api_if_srclist_host_address_get_next . . . 328
mrib6_mld_snmp_api_if_srclist_expiry_time_get_next . . . 329
MLD API Error Codes . . . 331
mrib6_mld_snmp_api_strerror . . . 332

CHAPTER 14 Layer 2 Multicast Routing Information Base API . . . 333
Overview . . . 333
How L2mribd Interacts with Protocols . . . 333
Data Structures . . . 334
l2mrib_master . . . 334
l2mrib_mcast . . . 335
l2mrib_bridge . . . 336
l2mrib_if . . . 336

Preface

This guide describes the application programming interface (API) for the Multicast Routing Information Base (MRIB) in ZebOS-XP.

Audience

This guide is intended for developers who write code to customize and extend MRIB.

Conventions

Table P-1 shows the conventions used in this guide.

Table P-1: Conventions

Convention	Description
Italics	Emphasized terms; titles of books
Note:	Special instructions, suggestions, or warnings
`monospaced type`	Code elements such as commands, functions, parameters, files, and directories

Contents

This guide contains these chapters:

- Chapter 1, *ZebOS-XP Multicast Protocols and MRIB*
- Chapter 2, *ZebOS-XP and Multicast Forwarder Interaction*
- Chapter 3, *Multiple Registration Protocols*
- Chapter 4, *Data Structures*
- Chapter 5, *MRIBv4 Command API*
- Chapter 6, *MRIBv6 Command API*
- Chapter 7, *Multicast Routing MIB API*
- Chapter 8, *MRIBv4 IGMP Command API*
- Chapter 9, *L2 IGMP Snooping Command API*
- Chapter 10, *MRIBv4 IGMP MIB API*
- Chapter 11, *MRIBv6 MLD Command API*
- Chapter 12, *L2 MLD Snooping Command API*
- Chapter 13, *MRIBv6 MLD MIB API*

- Chapter 14, *Layer 2 Multicast Routing Information Base API*

Related Documents

The following guides are related to this document:

- *Multicast Routing Information Base Command Reference*
- *Network Services Module Command Reference*
- *Network Services Module Developer Guide*
- *Installation Guide*
- *Architecture Guide*

Note: All ZebOS-XP technical manuals are available to licensed customers at http://www.ipinfusion.com/support/document_list.

Support

For support-related questions, contact support@ipinfusion.com.

Comments

If you have comments, or need to report a problem with the content, contact techpubs@ipinfusion.com.

CHAPTER 1 ZebOS-XP Multicast Protocols and MRIB

ZebOS-XP multicast protocol modules work with a common Multicast Routing Information Base (MRIB) in a system. Additionally, PIM-SM and PIM-DM assume that a unicast route lookup service is available in the system.

This chapter provides an overview of the ZebOS-XP multicast protocol modules and MRIB. It does not describe other generic services in the system, such as interface and IP address information.

Multicast Architecture Overview

ZebOS-XP multicast supports a common MRIB across all multicast protocols (both IPv4 and IPv6). The multicast protocols communicate with the MRIB, and the MRIB communicates with the Multicast Forwarder. The MRIB allows multiple multicast protocols to function simultaneously, supporting the following functions:

- Multicast Virtual Interface (VIF) management for IPv4 networks
- Multicast Interface (MIF) management for IPv6 networks
- Multicast Route Entry management for IPv4 and IPv6
- Multicast Forwarder Event handling for IPv4 and IPv6
- Multicast Forwarding Entry statistics for IPv4 and IPv6
- Multicast Tunnel management for IPv4 only
- Multiple Registration Protocol (MRPs)
- IGMP v2/v3 services
- MLD v1/v2 services

The MRIB also provides support for register packet generation and statistics event generation. It has the capability to process statistics messages and register messages.

MRIBd

The MRIB daemon (`mribd`) UNIX process maintains the L3 MRIB within ZebOS-XP to support the L3 multicast routing. The MRIB daemon does the following:

- Manages L2 and L3 MRIB
- Defines the MRIB IPC, which contains L3 multicast IPC messages
- Defines the address family independent MRIB component

Code Structure of MRIB

The ZebOS-XP library directory contains the L3 multicast sub-directory called `mcast` that holds the L3 multicast specific code. The following table shows a summary of the multicast library organization:

Table 1-1: Multicast library

Multicast Library	lib/mcast/		
Layer 3 IGMP library		mcast4/igmp/	
Layer 3 MLD library		mcast6/mld/	
MRIB library		mrib/	
MRIBv4 library			mrib4/
MRIBv6 library			mrib6/

MRIBd Directory

The MRIBd process source code is organized under the MRIBd directory. The following table shows a summary of the directory organization under MRIBd directory:

Table 1-2: MRIBd Directory

MRIBd	mribd/		
MRIBv4		mrib4/	
MRIBv4 IGMP			igmp/
MRIBv6		mrib6/	
MRIBv6 MLD			mld/

MRIB Services

MRIB provides the following services:

- Communicates with the Multicast Forwarding Information Base (MFIB) and acts as a proxy for the multicast protocol modules, both for programming, and handling events from, the MFIB.
- Maintains a database of interfaces on which multicast routing is enabled. The MRIB assigns a unique index to each of these interfaces and provides the index information to the multicast protocol modules.
- Maintains a database of (S,G) forwarding entries from the multicast protocol modules.
- Polls the MFIB for multicast forwarding statistics for each (S,G) entry.
- Maintains a keep-alive timer for an (S,G) entry, if requested by the protocol. If multicast traffic for the (S,G) entry stops (based on statistics polling), the MRIB notifies the multicast protocol module when the keep-alive timer expires.
- For PIM-SM, MRIB generates a PIM register message after the multicast protocol module acknowledges a WHOLEPKT from MFIB

Multicast-Related Modules

The following modules are used for the purpose of multicast routing in ZebOS-XP:

- Protocol modules such as `pimd` for multicast routing
- The `mribd` daemon for multicast RIB management
- NSM for VR/VRF and interface management
- HSL or kernel for multicast FIB management

Figure 1-1 shows a high level view of the process relationship and Inter-process Communications (IPCs) involved in multicast routing within ZebOS-XP.

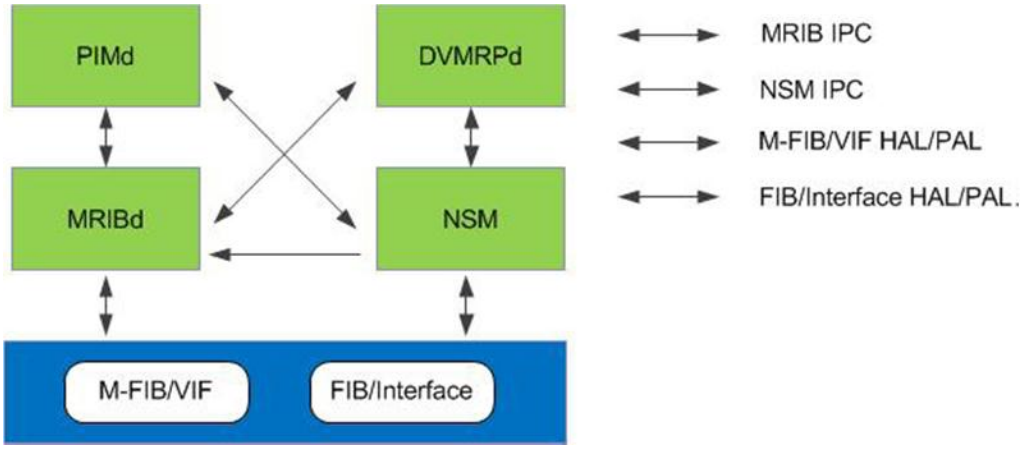

Figure 1-1: Modules Used in Multicast Routing and their Process Relationships

Multicast Protocol and MRIB Message Exchange

The following lists messages exchanged between a multicast protocol module and MRIB.

VIF add

Direction: multicast protocol module —> MRIB

This message is sent by an multicast protocol module when multicast routing is enabled on an interface.

VIF delete

Direction: multicast protocol module —> MRIB

This message is sent by an multicast protocol module when multicast routing is disabled on an interface.

Multicast route entry add

Direction: multicast protocol module —> MRIB

This message is sent by an multicast protocol module to the MRIB to add an (S,G) forwarding entry.

Multicast route entry delete

Direction: multicast protocol module —> MRIB

This message is sent by an multicast protocol module to the MRIB to delete an (S,G) forwarding entry.

Multicast forwarding statistics type

Direction: multicast protocol module —> MRIB

For a (S,G) entry, a multicast protocol module can request an immediate or timed statistics update from MRIB. For immediate statistics, MRIB notifies a multicast protocol module when the forwarding statistics of the (S,G) entry are updated after a poll. For timed statistics, MRIB maintains a keep-alive timer, and if forwarding statistics are not updated in the keep-alive timer interval, the MRIB notifies the multicast protocol module.

Cache Miss event

Direction: MRIB —> multicast protocol module

This event is sent by the MRIB to a multicast protocol module when it receives a Cache Miss event for an (S,G) from the MFIB.

Wrong incoming interface event

Direction: MRIB —> multicast protocol module

This event is sent by MRIB to a multicast protocol module when it receives a wrong incoming interface event for an (S,G) entry from MFIB

Whole packet notification

Direction: MRIB —> multicast protocol module

This event is sent by MRIB to a multicast protocol module when it receives a WHOLEPKT event for an (S,G) entry from MFIB.

Whole packet acknowledge

Direction: multicast protocol module —> MRIB

A multicast protocol module responds to a Whole packet notification from the with an ACK or NACK. MRIB generates a PIM-SM Register message for an ACK.

General Notifications

Direction: MRIB —> multicast protocol module

This message is for general notifications from the MRIB to a multicast protocol module. These notifications are listed below:

- VIF add/delete acknowledgement (for example, success, error)
- Multicast route entry add/delete acknowledgement (for example, success, error)
- Multicast routing start/stop
- Multicast route entry clear

IGMP

Multicast protocol modules rely on IGMP running as a separate module. These message are exchanged to and from a multicast protocol module and IGMP.

IGMP Local Membership update

Direction: IGMP —> multicast protocol module

This message carries information about local group membership updates on an interface from IGMP to a multicast protocol module.

IGMP Local Membership state refresh

Direction: multicast protocol module —> IGMP

A multicast protocol module can request a refresh of an interface local group membership from IGMP.

CHAPTER 2 ZebOS-XP and Multicast Forwarder Interaction

The MRIBd handles all interaction with the Multicast Forwarder in the TCP/IP stack. These interactions can be classified into two categories:

- Multicast forwarder programming
- Handling events from the multicast forwarder

This chapter describes these two types of interaction.

Note: This description applies to both IPv4 and IPv6 multicast forwarding. Minor differences between IPv4 and IPv6 multicast forwarder interactions are not described.

Multicast Forwarder Programming

The MRIBd acts as a proxy between the multicast routing protocols and the multicast forwarder . The multicast routing protocols send programming information to MRIBd, and MRIBd converts this information to the appropriate commands for the multicast forwarder. In a software-only multicast forwarding environment, MRIBd interacts with the multicast forwarder via ZebOS-XP Platform Abstraction Layer (PAL) APIs. For hardware multicast forwarders, MRIBd uses Hardware Abstraction Layer (HAL) APIs to program the multicast forwarder. In turn, HAL interacts with the Hardware Services Layer (HSL). HSL programs the hardware multicast forwarder and software multicast forwarder (used for slow-path forwarding) in the TCP/IP stack.

Multicast programming actions are described below.

Multicast Forwarding Start

This action starts multicast forwarding in the multicast forwarder.

Multicast Forwarding Stop

This action stops multicast forwarding in the multicast forwarder.

PIM Multicast Forwarding Start

This action tells the multicast forwarder to apply PIM-SM forwarding rules.

PIM Multicast Forwarding Stop

This action tells the multicast forwarder to stop applying PIM-SM forwarding rules.

Multicast Forwarding Start on an Interface

This action tells the multicast forwarder to start forwarding multicast traffic to and from a particular interface. Each interface enabled for multicast forwarding is assigned an index by ZebOS-XP. This index is passed to the multicast forwarder and is used to represent the interface when ZebOS-XP adds or removes a multicast forwarding entry.

Multicast Forwarding Stop on an Interface

This action tells the multicast forwarder to stop forwarding multicast traffic to and from a particular interface.

Multicast Forwarding Entry Add

This action adds an (S,G) multicast forwarding entry into the multicast forwarder. The incoming interface index, and a set of outgoing interface indices, represent the multicast distribution tree for an (S,G) entry.

Multicast Forwarding Entry Delete

This action deletes an (S,G) multicast forwarding entry from the multicast forwarder.

Multicast Forwarding Entr Statistics Get

This action retrieves the multicast forwarding statistics for an (S,G) entry from the multicast forwarder. The forwarding statistics are used by ZebOS-XP to determine multicast traffic liveliness.

Per-Interface Multicast Forwarding Statistics Get

This action retrieves the multicast forwarding statistics for an interface from the multicast forwarder.

Multicast Forwarder Events

Multicast forwarding is event driven, that is, multicast entries are added to the multicast forwarder based on events coming up from the multicast forwarder. MRIBd acts a proxy by relaying the events from the software multicast forwarder in the TCP/IP stack to the multicast routing protocols. All events from the multicast forwarder are generated based on incoming multicast packets. Therefore, even in hardware-based multicast forwarders, the events are generated from the software multicast forwarder in the stack, that is, events are processed in the slow path. These events are described below.

Cache miss event

This event is generated by the multicast forwarder when it does not have an (S,G) forwarding entry for an incoming multicast packet. Typically, the handling of this event in the multicast routing protocols results in the addition of a multicast forwarding entry in the multicast forwarder. The multicast packet incoming interface information is available in this event, so that multicast routing protocols can perform Reverse Path Forwarding (RPF) checks.

Wrong incoming interface event

This event is generated by the multicast forwarder when a multicast packet arrives on an interface that does not match the expected incoming interface of an (S,G) forwarding entry. This event is required by PIM-SM to perform Assert processing and switch from the shared (RP based) tree to the source tree. This event is rate controlled to prevent control plane overloading.

Whole packet event

When a multicast source starts generating traffic for a group, the first-hop PIM-SM router encapsulates a few initial multicast data packets to the RP. To do this, the PIM-SM routing protocol gets a whole multicast data packet from the multicast forwarder, so that it can be encapsulated. This is achieved by using a special logical interface, called the Register interface. At the first-hop router, PIM-SM adds an (S,G) forwarding entry with the Register interface in the outgoing interface set. When a multicast forwarder forwards a multicast packet on this (S,G) entry, it generates the whole packet event (containing the complete multicast packet) up to the control plane.

CHAPTER 3 Multiple Registration Protocols

This chapter describes Multiple Registration Protocols (MRPs) implementation within ZebOS-XP, including both Multiple MAC Registration Protocol (MMRP) and GARP Multiple Registration Protocol (GMRP) This section includes an overview of MRP, a list of MRP features, and a description of the supported MRP APIs.

Overview

By definition, MRP specifies the protocol, procedures, and managed objects required to support multiple registrations. This allows participants in an MRP application to register attributes with other participants in a bridged LAN. ZebOS-XP supports the following types of MRP

- Multiple MAC Registration Protocol (MMRP) manages group Media Access Control (MAC) addresses. In addition, MMRP improves the convergence time of the GMRP module.
- GARP Multicast Registration Protocol: GMRP provides multicast pruning and dynamic group membership for multicast traffic. A switch can be used to exchange multicast group information with other GMRP switches, prune unnecessary broadcast traffic, and dynamically create and manage multicast groups.

Architecture

An MRP application operates in networks where bridges support basic or extended filtering services. There can be a single MRP participant per port. Each MRP participant has an MRP application component, and an MRP Attribute Declaration (MAD) associated with each port. Information propagates among participants within a bridge by use of MRP Attribute Propagation (MAP) component. PDUs destined to a group MAC address specific to the MRP application will exchange the information among participants across the bridge. The MAD component has set of state machines running within the participant for which it is a component. The state machines define the set of declarations and the registrations of the attribute associated with its participant. There are two state machines for an attribute; one is the Registrar state machine and the second is the Applicant state machine.

The state machines differ depending on the topology or scenario, for example, shared media, point-to-point LAN, etc, where an MRP application is running to ensure optimal operation. Another component, called LeaveAll, is associated with each participant and generates a leaveall message once the leaveall timer run by the application has expired. To avoid data loops, when there is a transition of port role from alternate or root port to designated, the information maintained by the registrar of the alternate or root port is flushed, as stored in the state machine, and a leave message is sent to that port. However, when a port role transitions from designated to alternate or root port, no message is sent.

Functionality

By default, MRP allows participants in a MRP application to register with the other participants in a bridged LAN. MRP continues to use the GARP architecture. Its primary advantage is to continue the smooth operation of the bridges in large areas of provider-bridged network. However, GARP has limitations, including:

- For a point-to-point link, the message type used is the same as that for shared media, which results in the necessity of managing complexity comparable to shared media.
- Wherever there is a network failure, it affects the entire set of registrations.
- To transmit information about all 4096 possible VLANs, 11 frames need to be transmitted.
- Transmission of two JoinIn messages to avoid data loss is maintained at the time interval of the JoinTimer.

Features

To overcome the limitations of GARP, ZebOS-XP implements MRP according to the requirements of the IEEE 802.1AK(D4.0) specification. The following points highlight the features of MRP:

- An MRP application maintains one instance of an MRP applicant per port. Each MRP applicant has an application component and an MRP MAD component. Propagation of attributes between two participants of in the same application within a bridge is managed by an MRP Attribute Propagation (MAP) component. Each MAD maintains a registrar and applicant state machine, which is similar to that of GARP.
- An MRP application in a bridge uses a MAP context that defines the ports through which a declaration needs to be propagated.
- Since the nature of attribute propagation is store and forward, in a bridge running RSTP or MSTP, where role changes are faster, there is a risk of data loops. This situation is avoided by deleting the information held by the MADs registrar of the port whenever it transitions from root or alternate to designated forwarding.
- In the case of point-to-point links, only New (declare a new attribute), Join (re-declare an attribute), and Lv (withdraw the declaration) messages are allowed.
- Two messages for shared media — NewEmpty and NewJoin — are similar to JoinEmpty and JoinIn, but carry additional information needed to propagate topology change information. ChangingMember is an additional member state associated with the active member of an applicant that establishes that messages sent from this state indicate that there is a topology change associated with the attributes.
- A single LeaveAll state machine is maintained for every MRP participant, and it manages all events for all applicants and registrar state machine associated with the participant.
- Although MRP protocol exchanges can take place on all ports of the bridge, MRPPDUs are transmitted only through the port that forms the active topology of the bridge instance.
- In order to avoid information loops when there is any change in the port role (from Alternate or Root to Designated), information maintained with the registrar of that port is removed.

MRP Data Structures

The following are the data structures that support MRP, MMRP and MVRP functionality.

struct garp

This structure contains the list of function pointers to the various activities of join or leave indication, propagation, PDU transmit, VID of the corresponding application (MMRP).

struct garp_instance

This structure maintains both applications (MMRP) that contain the application of this instance and the count of the number of packets transmitted and received on this instance.

struct gid

This structure contains the structure to the GARP instance to which it is associated and the GID port structure where this instance is maintained. It also maintains the list of flags to indicate starting and running the applicant and registrar Join and Leave timers and the thread to all the timers.

struct gid_port

This structure contains the index and the timer value associated with a specific port.

struct gmrp_bridge

This structure contains the bridge structure with which the MMRP application is associated, the vlan_table of the MMRP instance, the function pointers to listener callbacks, and the GARP structure with which it is associated.

struct gmrp

This structure contains the MMRP bridge structure, the MRP instance for this instance and the VLAN ID. of this MMRP instance.

struct gmrp_port

This structure contains the MAD structure and the configuration flags for the port and is maintained on a per-port basis.

struct gmrp_port_instance

This structure contains the VLAND ID associated with the port for an instance of MMRP and the MAD structure.

struct gmrp_port_config

This structure contains the timer associated with the port.

struct gmrp_gmd_entry

This structure maintains group membership information, such as MAC address and flags.

struct gmrp_attr_entry_tbd

This structure maintains the attribute index entry for MMRP.

struct gmrp_gmd

This structure maintains the index manager of the attribute entry, the GMD entry table, and the attribute delete from the entry.

typedef enum gid_event

This structure identifies the event of a received PDU.

enum applicant_states

This structure contains the state of the applicant state machine.

garp_attribute_event

This structure contains the type of attribute.

struct mad_machine

This structure contains the state of the applicant and the registrar with respect to the attribute. This information is maintained on a per-port, per-attribute basis.

struct mad_states

This structure contains the MAD state of the applicant and the registrar.

struct garp

This structure contains the function pointer for the GARP instance callback functions.

API

The functions in this section are used with the GARP Multicast Registration Protocol (GMRP). Multiple MAC Registration Protocol (MMRP) structures are exactly the same as for GMRP.

gmrp_set_timer

This function sets the GMPR timer.

Syntax

```
s_int32_t
gmrp_set_timer (struct nsm_bridge_master *master,
                struct interface *ifp,
                const u_int32_t timer_type,
                const pal_time_t timer_value)
```

Input Parameters

`master`	Pointer to the NSM master.
`ifp`	Interface pointer.
`timer_type`	Choose one of these defined in the enum `garp_timers` in `nsm/L2/garp/garp_gid.c`:
	GARP_JOIN_TIMER
	GARP_LEAVE_TIMER
	GARP_LEAVE_ALL_TIMER
	GARP_LEAVE_CONF_TIMER
	GARP_LEAVEALL_CONF_TIMER
`timer_value`	Timer value in hundredths of a second.

Output Parameters

None

Return Values

RESULT_OK when the function succeeds

RESULT_ERROR when the function fails

gmrp_enable

This function enables GMRP on a bridge.

Syntax

```
s_int32_t
gmrp_enable (struct nsm_bridge_master *master,
             char *protocol,
             const char* const bridge_name)
```

Input Parameters

`master`	Pointer to the NSM master.
`protocol`	Type of protocol.
`bridge_name`	Name of the bridge.

Output Parameters

None

Return Values

RESULT_OK when the function succeeds

gmrp_disable

This function disables GMRP on a bridge.

Syntax

```
s_int32_t
gmrp_disable (struct nsm_bridge_master *master,
              char* bridge_name)
```

Input Parameters

`master`	Pointer to the NSM master.
`bridge_name`	Name of the bridge.

Output Parameters

None

Return Values

RESULT_OK when the function succeeds

RESULT_ERROR when the function fails

gmrp_enable_port

This function enables GMRP on a port.

Syntax

```
s_int32_t
gmrp_enable_port (struct nsm_bridge_master *master,
                  struct interface *ifp)
```

Input Parameters

`master`	Pointer to the NSM master.
`ifp`	Interface pointer.

Output Parameters

None

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

gmrp_disable_port

This function disables GMRP on a port.

Syntax

```
s_int32_t
gmrp_disable_port (struct nsm_bridge_master *master,
                   struct interface *ifp)
```

Input Parameters

`master`	Pointer to the NSM master
`ifp`	Interface pointer.

Output Parameters

None

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

gmrp_set_registration

This function sets GMRP registrations.

Syntax

```
s_int32_t
gmrp_set_registration (struct nsm_bridge_master *master,
                      struct interface *ifp, const u_int32_t registration_type)
```

Input Parameters

`master` Pointer to NSM master.

`ifp` Interface pointer.

`registration_type`

Choose from:

GID_EVENT_NORMAL_REGISTRATION
GID_EVENT_FIXED_REGISTRATION
GID_EVENT_FORBID_REGISTRATION

Output Parameters

None

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

gmrp_get_per_vlan_statistics_details

This function gets statistical data for GMRP.

Syntax

```
s_int32_t
gmrp_get_per_vlan_statistics_details (struct nsm_bridge_master *master,
                                      const char* const bridge_name,
                                      const u_int16_t vid,
                                      u_int32_t *receive_counters,
                                      u_int32_t *transmit_counters)
```

Input Parameters

`master` Pointer to NSM master.

`bridge_name` Name of the bridge.

`vid` VLAN ID to be configured.

Output Parameters

`receive_counter` Number of GMRP packets received.

`transmit_counter`

Number of GMRP packets transmitted.

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

gmrp_clear_all_statistics

This function clears all the received and transmitted GMRP statistics.

Syntax

```
s_int32_t
gmrp_clear_all_statistics (struct cli *cli)
```

Input Parameters

`cli`	CLI parameters.

Output Parameters

None

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

gmrp_clear_per_vlan_statistics

This function clears all the received and transmitted GMRP statistics per VLAN.

Syntax

```
s_int32_t
gmrp_clear_per_vlan_statistics (struct nsm_bridge_master *master,
                                const char* const bridge_name,
                                const u_int16_t vid)
```

Input Parameters

`master`	Pointer to the NSM master.
`bridge_name`	Name of the bridge.

Output Parameters

None

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

gmrp_set_fwd_all

This function sets GMRP to forward all.

Syntax

```
s_int32_t
gmrp_set_fwd_all (struct nsm_bridge_master *master,
                  struct interface *ifp, const u_int32_t event)
```

Input Parameters

`master`	Pointer to NSM master.
`ifp`	Interface pointer.
`event`	Event to perform: GID_EVENT_JOIN GID_EVENT_LEAVE

Output Parameters

None

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

gmrp_disable_instance

This function disables a bridge instance from a VLAN.

Syntax

```
s_int32_t
gmrp_disable_instance (struct nsm_bridge_master *master,
                       char* bridge_name, u_int16_t vid)
```

Input Parameters

`master`	Pointer to the NSM master.
`bridge_name`	Name of the bridge.
`vid`	VLAN ID to be configured.

Output Parameters

None

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

gmrp_enable_instance

This function enables a bridge instance on a VLAN.

Syntax

```
s_int32_t
gmrp_enable_instance (struct nsm_bridge_master *master,
                      char *protocol, char* bridge_name, u_int16_t vid)
```

Input Parameters

`master`	Pointer to the NSM master.
`bridge_name`	Name of the bridge.
`vid`	VLAN ID to be configured.

Output Parameters

None

Return Values

CLI_SUCCESS when the function succeeds

CLI_ERROR when the function fails

CHAPTER 4 Data Structures

This chapter describes the data structures in MRIB.

Common Data Structures

The following data structures are common for all ZebOS-XP protocols and are used in MRIBd functions:

- `interface`
- `lib_globals`

See the *Common Data Structures Developer Guide* for a description of these data structures.

Multicast Data Structures

This section describes the MRIBd data structures used in this guide. Each section contains the actual data structures as well as the header file that contains the actual definition.

mrib_snmp_api_vif_index

This structure is used to provide the interface index information to access the ipMRoutInterfaceTable entries.

It resides in the `mribd/mrib_snmp_api.h` file

```
struct mrib_snmp_api_vif_index
{
  u_int32_t len;
  u_int32_t ip_version;
  u_int32_t ifindex;
};
```

mrib_snmp_api_mrt_index

This structure is used to provide the multicast route index information to access the ipMRouteTable entries.

```
struct mrib_snmp_api_mrt_index
{
  unsigned int len;
  u_int32_t group_addr_type;
  u_int32_t group_prefix_len;
  u_int32_t source_addr_type;
  u_int32_t source_prefix_len;
#ifdef HAVE_MRIB_IPV4
  struct pal_in4_addr group;
  struct pal_in4_addr source;
#endif /* HAVE_MRIB_IPV4 */
#ifdef HAVE_MRIB_IPV6
```

```
  struct pal_in6_addr group6;
  struct pal_in6_addr source6;
#endif /* HAVE_MRIB_IPV6 */
};
```

mrib_snmp_api_nh_index

This structure is used to provide the nexthop index information to access to the ipMRouteNexthopTable.

It resides in the `mribd/mrib_snmp_api.h` file

```
struct mrib_snmp_api_nh_index
{
  unsigned int len;
  u_int32_t group_addr_type;
  struct pal_in4_addr group;
#ifdef HAVE_MRIB_IPV6
  struct pal_in6_addr group6;
#endif /* HAVE_MRIB_IPV6 */
  u_int32_t group_prefix_len;

  u_int32_t source_addr_type;
  struct pal_in4_addr source;
#ifdef HAVE_MRIB_IPV6
  struct pal_in6_addr source6;
#endif /* HAVE_MRIB_IPV6 */
  u_int32_t source_prefix_len;

  u_int32_t ifindex;
  u_int32_t addr_type;
  struct pal_in4_addr addr;
#ifdef HAVE_MRIB_IPV6
  struct pal_in6_addr addr6;
#endif /* HAVE_MRIB_IPV6 */
};
```

igmp_group_rec

This structure provides the interface group record information, which is the key to track the group membership on the local interface.

It resides in the `lib/igmp/igmp_struct.h` file.

```
/* IGMP Interface Group Record */
struct igmp_group_rec
{
  /* IGMP Group Record Owning P-Trie Node */
  struct ptree_node *igr_owning_pn;

  /* IGMP Group Record Owning IGMP IF */
  struct igmp_if *igr_owning_igif;

  /* IGMP Group Record Last Reporting Host */
  struct pal_in4_addr igr_last_reporter;

  /* IGMP Group Record Uptime  */
  pal_time_t igr_uptime;

  /* IGMP Group Record Liveness Timer Value */
  u_int32_t v_igr_liveness;

  /* IGMP Group Record Liveness Timer Thread */
  struct thread *t_igr_liveness;

  /*
   * IGMP Group Record Filter-Mode State
   * (variable overloaded for both Router & Host FSMs)
   */
  enum igmp_filter_mode_state igr_filt_mode_state;

  /* IGMP Group Record Source-List-A P-Trie */
  struct ptree *igr_src_a_tib;

  /* IGMP Group Record Source-List-B P-Trie */
  struct ptree *igr_src_b_tib;

  /* IGMP Group Record Source-List-A Count */
  u_int16_t igr_src_a_tib_count;

  /* IGMP Group Record Source-List-B Count */
  u_int16_t igr_src_b_tib_count;

  /* IGMP Group Ver1 Host Present Timer Thread */
  struct thread *t_igr_v1_host_present;

  /* IGMP Group Ver2 Host Present Timer Thread */
  struct thread *t_igr_v2_host_present;
```

```
/* IGMP Group Query Re-transmit  Count */
u_int16_t igr_rexmit_group_lmqc;

/* IGMP Group-Source Query Re-transmit Count */
u_int16_t igr_rexmit_group_source_lmqc;

/* IGMP Grp/Grp-Src Report Re-transmit Host Rec Type */
enum igmp_hst_rec_type igr_rexmit_hrt;

/*
 * IGMP Group Query Re-transmission Timer Thread
 * (variable overloaded for sending Grp-Report (Host-side))
 */
struct thread *t_igr_rexmit_group;

/*
 * IGMP Group-Source Query Re-transmission Timer Thread
 * (variable overloaded for sending Grp-Src-Report (Host-side))
 */
struct thread *t_igr_rexmit_group_source;

/*
 *IGMP Group Query Re-transmission Timer Thread
 *(variable overloaded for sending Grp-Report (Join-group))
 */
struct thread *t_igr_join_group;

/*
 * IGMP Group-Source Query Re-transmission TIB
 * (variable overloaded for Rexmit Grp-Src-Report (Host-side))
 */
struct ptree *igr_rexmit_srcs_tib;

/* IGMP Group Source ALLOW-NEW_SRCS Rexmit TIB */
struct ptree *igr_rexmit_allow_tib;

/* IGMP Group Source BLOCK-OLD_SRCS Rexmit TIB */
struct ptree *igr_rexmit_block_tib;

/*
 * IGMP Group-Source Query Re-transmission TIB Count
 * (variable overloaded for Rexmit Grp-Src-Report (Host-side))
 */
u_int16_t igr_rexmit_srcs_tib_count;

/* IGMP Group Source ALLOW-NEW_SRCS TIB Count */
u_int16_t igr_rexmit_srcs_allow_tib_count;

/* IGMP Group Source BLOCK-OLD_SRCS TIB Count */
```

```
  u_int16_t igr_rexmit_srcs_block_tib_count;

  /* IGMP Group Record Status-Flags */
  u_int16_t igr_sflags;
#define IGMP_IGR_SFLAG_COMPAT_V1                (1 << 0)
#define IGMP_IGR_SFLAG_COMPAT_V2                (1 << 1)
#define IGMP_IGR_SFLAG_COMPAT_V3                (1 << 2)
#define IGMP_IGR_SFLAG_REPORT_PENDING           (1 << 3)
#define IGMP_IGR_SFLAG_MFC_PROGMED              (1 << 4)
#define IGMP_IGR_SFLAG_STATE_REFRESH            (1 << 5)
#define IGMP_IGR_SFLAG_STATIC                   (1 << 6)
#define IGMP_IGR_SFLAG_DYNAMIC                  (1 << 7)
#define IGMP_IGR_SFLAG_JOIN                     (1 << 8)
  u_int32_t igr_cflags;
#define IGMP_IGR_CFLAG_STATIC_GROUP             (1 << 0)
#define IGMP_IGR_CFLAG_STATIC_GROUP_SOURCE      (1 << 1)
#define IGMP_IGR_CFLAG_STATIC_SOURCE_SSM_MAP    (1 << 2)
#define IGMP_IGR_CFLAG_STATIC_GROUP_IF_NAME     (1 << 3)
#define IGMP_IGR_CFLAG_LOCAL_JOIN_GROUP         (1 << 4)
};
```

igmp_instance

This structure is the top structure to keep track of the IGMP state information per VRF instance.

It resides in the `lib/mcast/mcast4/igmp/mcast4_igmp.h` file

```
/* IGMP Instance Structure */
struct igmp_instance
{
  /* Owning module's Library globals */
  struct lib_globals *igi_lg;

  /* Owning Library VRF structure */
  struct ipi_vrf *igi_owning_ivrf;

  /* IGMP Svc Registrations List */
  struct list igi_svc_reg_lst;

  /* IGMP SSM-Map Static List */
  struct list igi_ssm_map_static_lst;

  /* IGMP Interfaces AVL Tree */
  struct avl_tree *igi_if_tree;

  /* IGMP Input buffer */
  struct stream *igi_i_buf;

  /* IGMP Output buffer */
  struct stream *igi_o_buf;
```



```
  /* IGMP Tunnel Interface Get */
  igmp_cback_tunnel_get_t igi_cback_tunnel_get;

  /* IGMP Instance wide limit */
  u_int32_t igi_limit;

  /* IGMP Instance wide limit exception ACL */
  u_int8_t *igi_limit_except_alist;

  /* IGMP Instance wide G-Recs Count */
  u_int32_t igi_num_grecs;

  /* IGMP Join group socket */
  pal_sock_handle_t igmp_join_sock;
  pal_sock_handle_t igmp_join_read_sock;
  pal_sock_handle_t igmp_join_write_sock;

  /* IGMP join group read thread */
  struct thread *t_igmp_join_read;

  /* IGMP well-known muticast address in Network-order */
  struct pal_in4_addr igi_allhosts;
  struct pal_in4_addr igi_allrouters;
  struct pal_in4_addr igi_igmp_v3routers;
  struct pal_in4_addr igi_in4any_addr;

  /* IGMP Instance Configuration Flags */
  u_int16_t igi_cflags;
#define IGMP_INST_CFLAG_LIMIT_GREC          (1 << 0)
#define IGMP_INST_CFLAG_SNOOP_DISABLED      (1 << 1)
#define IGMP_INST_CFLAG_SSM_MAP_DISABLED    (1 << 2)
#define IGMP_INST_CFLAG_SSM_MAP_STATIC      (1 << 3)

  /* IGMP Instance Status Flags */
  u_int16_t igi_sflags;
#define IGMP_INST_SFLAG_SNOOP_ENABLED       (1 << 0)
#define IGMP_INST_SFLAG_L3_ENABLED          (1 << 1)

  /* IGMP Instance Debug Flags */
  u_int32_t igi_conf_dbg_flags;
  u_int32_t igi_term_dbg_flags;
#define IGMP_INST_DBG_DECODE                (1 << 0)
#define IGMP_INST_DBG_ENCODE                (1 << 1)
#define IGMP_INST_DBG_EVENTS                (1 << 2)
#define IGMP_INST_DBG_FSM                   (1 << 3)
#define IGMP_INST_DBG_TIB                   (1 << 4)
};
```

mld_instance

This structure is the top structure to keep track of the MLD state information per VRF instance.

It resides in the following files:

- lib/mcast/mcast6/mld/mcast6_mld.h
- lib/mld/mld_struct.h

```
/* MLD Instance Structure */
struct mld_instance
{
  /* Owning module's Library globals */
  struct lib_globals *mli_lg;

  /* Owning Library VRF structure */
  struct ipi_vrf *mli_owning_ivrf;

  /* MLD Svc Registrations List */
  struct list mli_svc_reg_lst;

  /* MLD SSM-Map Static List */
  struct list mli_ssm_map_static_lst;

  /* MLD Interfaces AVL Tree */
  struct avl_tree *mli_if_tree;

  /* MLD Input buffer */
  struct stream *mli_i_buf;

  /* MLD Output buffer */
  struct stream *mli_o_buf;

  /* MLD Tunnel Interface Get */
  mld_cback_tunnel_get_t mli_cback_tunnel_get;

  /* MLD Instance wide limit */
  u_int32_t mli_limit;

  /* MLD Instance wide limit exception ACL */
  u_int8_t *mli_limit_except_alist;

  /* MLD Instance wide G-Recs Count */
  u_int32_t mli_num_grecs;

  /* MLD IPv6 Wild-Card Address in Network-order */
  struct pal_in6_addr mli_in6any_addr;

  /* MLD IPv6 Wild-Card Multicast Address in Network-order */
  struct pal_in6_addr mli_in6wc_addr;
```

```
  /* MLD IPv6 All-Nodes Multicast Address in Network-order */
  struct pal_in6_addr mli_allnodes_addr;

  /* MLD IPv6 All-Routers Multicast Address in Network-order */
  struct pal_in6_addr mli_allrouters_addr;

  /* MLD IPv6 MLDv2 Routers Multicast Address in Network-order */
  struct pal_in6_addr mli_mldv2routers_addr;

  /* MLD Instance Configuration Flags */
  u_int16_t mli_cflags;
#define MLD_INST_CFLAG_LIMIT_GREC           (1 << 0)
#define MLD_INST_CFLAG_SNOOP_DISABLED       (1 << 1)
#define MLD_INST_CFLAG_SSM_MAP_DISABLED     (1 << 2)
#define MLD_INST_CFLAG_SSM_MAP_STATIC       (1 << 3)

  /* MLD Instance Status Flags */
  u_int16_t mli_sflags;
#define MLD_INST_SFLAG_SNOOP_ENABLED        (1 << 0)
#define MLD_INST_SFLAG_L3_ENABLED           (1 << 1)

  /* MLD Module Debug Flags */
  u_int32_t mli_conf_dbg_flags;
  u_int32_t mli_term_dbg_flags;
#define MLD_INST_DBG_DECODE            (1 << 0)
#define MLD_INST_DBG_ENCODE            (1 << 1)
#define MLD_INST_DBG_EVENTS            (1 << 2)
#define MLD_INST_DBG_FSM               (1 << 3)
#define MLD_INST_DBG_TIB               (1 << 4)
};
```

mld_group_rec

This structure provide the interface group record information, which is the key to track the group membership on the local interface.

It resides in the l`ib/mcast/mcast6/mld/mcast6_mld.h` file

```
/* MLD Interface Group Record */
struct mld_group_rec
{
  /* MLD Group Record Owning P-Trie Node */
  struct ptree_node *mgr_owning_pn;

  /* MLD Group Record Owning MLD IF */
  struct mld_if *mgr_owning_mlif;

  /* MLD Group Record Last Reporting Host */
  struct pal_in6_addr mgr_last_reporter;

  /* MLD Group Record Uptime  */
  pal_time_t mgr_uptime;

  /* MLD Group Record Liveness Timer Value */
  u_int32_t v_mgr_liveness;

  /* MLD Group Record Liveness Timer Thread */
  struct thread *t_mgr_liveness;

  /*
   * MLD Group Record Filter-Mode State
   * (variable overloaded for both Router & Host FSMs)
   */
  enum mld_filter_mode_state mgr_filt_mode_state;
 /* MLD Group Record Source-List-A P-Trie */
  struct ptree *mgr_src_a_tib;

  /* MLD Group Record Source-List-B P-Trie */
  struct ptree *mgr_src_b_tib;

  /* MLD Group Record Source-List-A Count */
  u_int16_t mgr_src_a_tib_count;

  /* MLD Group Record Source-List-B Count */
  u_int16_t mgr_src_b_tib_count;

  /* MLD Group Ver1 Host Present Timer Thread */
  struct thread *t_mgr_v1_host_present;

  /* MLD Group Query Re-transmit  Count */
  u_int16_t mgr_rexmit_group_lmqc;
```

```
  /* MLD Group-Source Query Re-transmit Count */
  u_int16_t mgr_rexmit_group_source_lmqc;

  /* MLD Grp/Grp-Src Report Re-transmit Host Rec Type */
  enum mld_hst_rec_type mgr_rexmit_hrt;

  /*
   * MLD Group Query Re-transmission Timer Thread
   * (variable overloaded for sending Grp-Report (Host-side))
   */
  struct thread *t_mgr_rexmit_group;

  /*
   * MLD Group-Source Query Re-transmission Timer Thread
   * (variable overloaded for sending Grp-Src-Report (Host-side))
   */
  struct thread *t_mgr_rexmit_group_source;

  /*
   * MLD Group-Source Query Re-transmission TIB
   * (variable overloaded for Rexmit Grp-Src-Report (Host-side))
   */
  struct ptree *mgr_rexmit_srcs_tib;

  /* MLD Group Source ALLOW-NEW_SRCS Rexmit TIB */
  struct ptree *mgr_rexmit_allow_tib;

  /* MLD Group Source BLOCK-OLD_SRCS Rexmit TIB */
  struct ptree *mgr_rexmit_block_tib;

  /*
   * MLD Group-Source Query Re-transmission TIB Count
   * (variable overloaded for Rexmit Grp-Src-Report (Host-side))
   */
  u_int16_t mgr_rexmit_srcs_tib_count;

  /* MLD Group Source ALLOW-NEW_SRCS TIB Count */
  u_int16_t mgr_rexmit_srcs_allow_tib_count;

  /* MLD Group Source BLOCK-OLD_SRCS TIB Count */
  u_int16_t mgr_rexmit_srcs_block_tib_count;

  /* MLD Group Record Status-Flags */
  u_int16_t mgr_sflags;
#define MLD_MGR_SFLAG_COMPAT_V1                 (1 << 0)
#define MLD_MGR_SFLAG_COMPAT_V2                 (1 << 1)
#define MLD_MGR_SFLAG_REPORT_PENDING            (1 << 2)
#define MLD_MGR_SFLAG_MFC_PROGMED               (1 << 3)
#define MLD_MGR_SFLAG_STATE_REFRESH             (1 << 4)
#define MLD_MGR_SFLAG_STATIC                    (1 << 5)
```

```
#define MLD_MGR_SFLAG_DYNAMIC                   (1 << 6)
  u_int32_t mgr_cflags;
#define MLD_MGR_CFLAG_STATIC_GROUP              (1 << 0)
#define MLD_MGR_CFLAG_STATIC_GROUP_SOURCE       (1 << 1)
#define MLD_MGR_CFLAG_STATIC_SOURCE_SSM_MAP     (1 << 2)
#define MLD_MGR_CFLAG_STATIC_GROUP_IF_NAME      (1 << 3)
};
```

l2mrib_mcast

This data structure in l2mribd.h maintains the details of IGMP/MLD snooping such as IGMP/MLD instances, input/output buffer, svc registration ID any many more.

```
/struct l2mrib_mcast
{
struct l2mrib_master *l2mm;
/* Packet Input/Output buffer */
struct stream *iobuf;
/* Packet Output buffer */
struct stream *obuf;
#ifdef HAVE_IGMP_SNOOP
/* IGMP Instance */
struct igmp_instance *igmp_inst;
/* IGMP L2 Service Registration ID */
void *igmp_svc_reg_id;
#endif
#ifdef HAVE_MLD_SNOOP
/* MLD Instance */
struct mld_instance *mld_inst;
/* MLD L2 Service Registration ID */
void *mld_svc_reg_id;
#endif
enum
{
L2MRIB_UNKNOWN_MCAST_FLOOD = 0,
L2MRIB_UNKNOWN_MCAST_DISCARD = 1,
}l2mrib_unknown_mcast;
```

l2mrib_master

This data structure in l2mribd.h holds an L2 related information

It resides in the l`ib/mcast/mcast6/mld/mcast6_mld.h` file

```
/struct l2mrib_master
{
struct ipi_vr *vr;
struct lib_globals *zg;
struct l2mrib_mcast *l2mcast;
struct list *mcast_bridge_list;
struct list *bridge_config; /* struct br_config*/
u_char config_flag;
#define CONFIG_IGMP_SNOOP_DISABLED (1<<0)
#define CONFIG_MLD_SNOOP_DISABLED (1<<1)
#ifdef HAVE_DISABLE_IGMP_SNOOP
#define CONFIG_IGMP_SNOOP_ENABLED (1<<2)
#endif /* HAVE_DISABLE_IGMP_SNOOP */
#ifdef HAVE_DISABLE_MLD_SNOOP
#define CONFIG_MLD_SNOOP_ENABLED (1<<3)
#endif /* HAVE_DISABLE_MLD_SNOOP */
};
```

L2mrib_bridge

This structure in l2mribd.h maintains all the bridge-related information, received from NSM, from the messages.

```
struct l2mrib_bridge
{
struct l2mrib_master *l2mm;
u_int8_t bridge_name[L2MRIB_BRIDGE_NAME_LEN+1];
u_int8_t bridge_type;
u_int8_t is_enabled;
struct avl_tree *port_list;
struct avl_tree *br_inst_list;
struct avl_tree *vlan_table;
struct avl_tree *snoop_entry;
struct thread *t_snoop_entry_send;
};
```

L2mrib_if

This structure in l2mribd.h maintains interface related information, updated by NSM.

```
struct l2mrib_if
{
  struct l2mrib_bridge      *br;

  struct l2mrib_port        *l2port;

  struct l2mrib_vlan        *vlan;

#ifdef HAVE_IGMP_SNOOP
  struct ptree *igmpsnp_gmr_tib;
#endif

#ifdef HAVE_MLD_SNOOP
  struct ptree *mldsnp_gmr_tib;
#endif

  u_char if_state;
#define L2MRIB_IF_DEFAULT                       (1 << 0)
#define L2MRIB_IF_ENABLED                       (1 << 1)
};
```

CHAPTER 5 MRIBv4 Command API

This chapter describes the Multicast Routing Information Base IPv4 (MRIBv4) command API.

API

Include File

To call the functions in this chapter, you must include `mribd/mrib4/mrib4_api.h`.

mrib4_api_multicast_routing_set

This function starts up the L3 IPv4 multicast routing on the router through the MRIBd process.

This function implements the `ip multicast-routing` command.

Syntax

```
int
mrib4_api_multicast_routing_set (u_int32_t vr_id, vrf_id_t vrf_id)
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_multicast_routing_unset

This function stops the L3 IPv4 multicast routing on the router through the MRIBd process.

This function is used to implement the `no ip multicast-routing` command.

Syntax

```
int
mrib4_api_multicast_routing_unset (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_rt_limit_thresh_set

This function sets the threshold of the route in the multicast route entries in the MRIB/MFIB.

This function is used to implement the `ip multicast route-limit` command.

Syntax

```
int
mrib4_api_rt_limit_thresh_set (u_int32_t vr_id, vrf_id_t vrf_id,u_int32_t rt_limit,
u_int32_t rt_thresh);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`rt_limit`	Route-limit number. This is the number of multicast routes that can be added to a multicast routing table
`rt_thresh`	Threshold value at which to generate a warning message

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_RT_THRESH_EXCEED_RT_LIMIT when the route threshold exceeds configured route limit

MRIB_API_SET_ERR_RT_LIMIT_EXCEED_RTS when the route limit exceeds the current number of routes

mrib4_api_rt_limit_thresh_unset

This function resets the threshold of the multicast route entries in the MRIB/MFIB.

This function is used to implement the `no ip multicast route-limit thresh` command.

Syntax

```
int
mrib4_api_rt_limit_thresh_unset (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_rt_limit_set

This function sets the limit of the multicast route entries in the MRIB/MFIB.

This function is used to implement the `ip multicast route-limit` command.

Syntax

```
int
mrib4_api_rt_limit_set (u_int32_t vr_id, vrf_id_t vrf_id, u_int32_t rt_limit);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`rt_limit`	Route-limit number. This is the number of multicast routes that can be added to a multicast routing table

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_RT_LIMIT_EXCEED_RTS when the route limit exceeds the current number of routes

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_vif_ttl_threshold_set

This function sets the multicast forwarding Time To Live (TTL) threshold value to the interface, which filters the multicast data packet. The interface has the greater TTL.

This function is used to implement the `no ip multicast ttl-threshold` command.

Syntax

```
int
mrib4_api_vif_ttl_threshold_set (const u_int32_t vr_id, const char *ifname,
                                 const u_char ttl);
```

Input Parameters

`vr_id`	Virtual router ID
`ifname`	Name of the interface

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VALUE when an invalid input value is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_vif_ttl_threshold_unset

This function resets the multicast TTL forwarding value to the default setting.

This function is used to implement the `no ip multicast ttl-threshold` command.

Syntax

```
int
mrib4_api_vif_ttl_threshold_unset (const u_int32_t vr_id, const char *ifname);
```

Input Parameters

`vr_id`	Virtual router ID
`ifname`	Name of the interface

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VALUE when an invalid input value is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_VIF_NOT_EXIST when a virtual interface structure (VIF) does not exists

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_clear_mroute_all

This function clears the multicast entry in the MRIB/MFIB.

This function is used to implement the `clear ip mroute *` command.

Syntax

```
int
mrib4_api_clear_mroute_all (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_clear_mroute_g

This function clears the multicast route entries, which has the specified group value in the MRIB/MFIB.

This function is used to implement the `clear ip mroute A.B.C.D` command.

Syntax

```
int
mrib4_api_clear_mroute_g (u_int32_t vr_id, vrf_id_t vrf_id,
                          struct pal_in4_addr *grp)
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`grp`	Group IP address

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_INVALID_GROUP_ADDRESS when an invalid group IP address is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_clear_mroute_sg

This function clears the multicast route entry, which matches the specified source (unicast) address and the group (multicast) address (S,G) entry.

This function is used to implement the `clear ip mroute A.B.C.D A.B.C.D` command.

Syntax

```
int
mrib4_api_clear_mroute_sg (u_int32_t vr_id, vrf_id_t vrf_id,
                           struct pal_in4_addr *src, struct pal_in4_addr *grp);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`src`	Source IP address
`grp`	Group IP address

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_INVALID_GROUP_ADDRESS when an invalid group IP address is used

MRIB_API_SET_ERR_INVALID_SOURCE_ADDRESS when an invalid source address is used

MRIB_API_SET_ERR_MRT_NOT_EXIST when the IP multicast route does not exists

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_clear_mroute_stats_all

This function clears the IPv4 multicast statistics from the MRIB/MFIB.

This function is used to implement the `clear ip mroute statistics *` command.

Syntax

```
int
mrib4_api_clear_mroute_stats_all (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_clear_mroute_stats_g

This function clears the multicast statics, which matches to the specific group entry.

This function is used to implement the `clear ip mroute statistics A.B.C.D` command.

Syntax

```
int
mrib4_api_clear_mroute_stats_g (u_int32_t vr_id, vrf_id_t vrf_id,
                                struct pal_in4_addr *grp);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`grp`	Group IP address

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_INVALID_GROUP_ADDRESS when an invalid group IP address is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_clear_mroute_stats_sg

This function clears the multicast statistics, which matches the specified (S, G) entry.

This function is used to implement the `clear ip mroute statistics A.B.C.D A.B.C.D` command.

Syntax

```
int
mrib4_api_clear_mroute_stats_sg (u_int32_t vr_id, vrf_id_t vrf_id,
                                 struct pal_in4_addr *src,
                                 struct pal_in4_addr *grp);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`src`	Source IP address
`grp`	Group IP address

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_INVALID_GROUP_ADDRESS when an invalid group IP address is used

MRIB_API_SET_ERR_INVALID_SOURCE_ADDRESS when an invalid source address is used

MRIB_API_SET_ERR_MRT_NOT_EXIST when the IP multicast route does not exists

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_all_set

This function enables debugging for all MRIBd events.

This function is used to implement the `debug ip mrib all` command.

Syntax

```
int
mrib4_api_debug_all_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_all_unset

This function disables debugging for all MRIBd events.

This function is used to implement the `no debug ip mrib all` command.

Syntax

```
int
mrib4_api_debug_all_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_event_set

This function enables debugging of MRIB events.

This function is used to implement the `debug ip mrib event` command.

Syntax

```
int
mrib4_api_debug_event_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_event_unset

This function disables debugging of MRIB events.

This function is used to implement the `no debug ip mrib event` command.

Syntax

```
int
mrib4_api_debug_event_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_vif_set

This function enables MRIB virtual interface structure (VIF) debugging.

This function is used to implement the `debug ip mrib vif` command.

Syntax

```
int
mrib4_api_debug_vif_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_vif_unset

This function disables MRIB VIF debugging.

This function is used to implement the `no debug ip mrib vif` command.

Syntax

```
int
mrib4_api_debug_vif_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_mrt_set

This function enables MRIB Multicast Route Table (MRT) debugging.

This function is used to implement the `debug ip mrib mrt` command.

Syntax

```
int
mrib4_api_debug_mrt_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_mrt_unset

This function disables MRIB MRT debugging.

This function is used to implement the `no debug ip mrib mrt` command.

Syntax

```
int
mrib4_api_debug_mrt_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_stats_set

This function enables MRIB statistics debugging.

This function is used to implement the `debug ip mrib stats` command.

Syntax

```
int
mrib4_api_debug_stats_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_stats_unset

This function disables MRIB statistics debugging.

This function is used to implement the `no debug ip mrib stats` command.

Syntax

```
int
mrib4_api_debug_stats_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_fib_msg_set

This function enables MRIB forwarding information base (FIB) message debugging.

This function is used to implement the `debug ip mrib fib-msg` command.

Syntax

```
int
mrib4_api_debug_fib_msg_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_fib_msg_unset

This function disables MRIB FIB message debugging.

This function is used to implement the `no debug ip mrib fib-msg` command.

Syntax

```
int
mrib4_api_debug_fib_msg_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_register_msg_set

This function enables MRIB register message debugging.

This function is used to implement the `debug ip mrib register-msg` command.

Syntax

```
int
mrib4_api_debug_register_msg_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_register_msg_unset

This function disables MRIB register message debugging.

This function is used to implement the `no debug ip mrib register-msg` command.

Syntax

```
int
mrib4_api_debug_register_msg_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_nsm_msg_set

This function enables MRIB NSM message debugging.

This function is used to implement the `debug ip mrib nsm-msg` command.

Syntax

```
int
mrib4_api_debug_nsm_msg_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_nsm_msg_unset

This function disables MRIB NSM message debugging.

This function is used to implement the `no debug ip mrib nsm-msg` command.

Syntax

```
int
mrib4_api_debug_nsm_msg_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_mrib_msg_set

This function enables MRIB MRIB message debugging.

This function is used to implement the `debug ip mrib mrib-msg` command.

Syntax

```
int
mrib4_api_debug_mrib_msg_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_mrib_msg_unset

This function disables MRIB MRIB message debugging.

This function is used to implement the `no debug ip mrib mrib-msg` command.

Syntax

```
int
mrib4_api_debug_mrib_msg_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_mtrace_set

This function enables MRIB mtrace debugging.

This function is used to implement the `debug ip mrib mtrace` command.

Syntax

```
int
mrib4_api_debug_mtrace_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_mtrace_unset

This function disables MRIB mtrace debugging.

This function is used to implement the `no debug ip mrib mtrace` command.

Syntax

```
int
mrib4_api_debug_mtrace_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_mtrace_detail_set

This function enables MRIB mtrace detailed debugging.

This function is used to implement the `debug ip mrib mtrace-detail` command.

Syntax

```
int
mrib4_api_debug_mtrace_detail_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib4_api_debug_mtrace_detail_unset

This function disables MRIB mtrace detailed debugging.

This function is used to implement the `no debug ip mrib mtrace-detail` command.

Syntax

```
int
mrib4_api_debug_mtrace_detail_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

CHAPTER 6 MRIBv6 Command API

This chapter describes the Multicast Routing Information Base IPv6 (MRIBv6) command API functions.

API

Include File

To call the functions in this chapter, you must include `mribd/mrib6/mrib6_api.h`.

mrib6_api_multicast_routing_set

This function starts up the L3 IPv6 multicast routing on the router through the MRIBd process.

This function is used to implement the `ipv6 multicast-routing` command.

Syntax

```
int
mrib6_api_multicast_routing_set (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_multicast_routing_unset

This function stops the L3 IPv6 multicast routing on the router through the MRIBd process.

This function is used to implement the `no ipv6 multicast-routing` command.

Syntax

```
int
mrib6_api_multicast_routing_unset (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_rt_limit_thresh_set

This function sets the threshold of the route in the multicast route entries in the MRIB/MFIB.

This function is used to implement the `ipv6 multicast route-limit <1-2147483647> <1-2147483647>` command.

Syntax

```
int
mrib6_api_rt_limit_thresh_set (u_int32_t vr_id, vrf_id_t vrf_id,
                               u_int32_t rt_limit, u_int32_t rt_thresh);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`rt_limit`	Route-limit number. This is the number of multicast routes that can be added to a multicast routing table
`rt_thresh`	Threshold value at which to generate a warning message

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_RT_THRESH_EXCEED_RT_LIMIT when the route threshold exceeds configured route limit

MRIB_API_SET_ERR_RT_LIMIT_EXCEED_RTS when the route limit exceeds the current number of routes

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_rt_limit_thresh_unset

This function resets the threshold of the multicast route entries in the MRIB/MFIB.

This function is used to implement the `no ipv6 multicast route-limit` command.

Syntax

```
int
mrib6_api_rt_limit_thresh_unset (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_rt_limit_set

This function sets the limit of the multicast route entries in the MRIB/MFIB.

This function is used to implement the `ipv6 multicast route-limit <1-2147483647>` command.

Syntax

```
int
mrib6_api_rt_limit_set (u_int32_t vr_id, vrf_id_t vrf_id, u_int32_t rt_limit);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`rt_limit`	Route-limit number. This is the number of multicast routes that can be added to a multicast routing table

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_RT_LIMIT_EXCEED_RTS when the route limit exceeds the current number of routes

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_clear_mroute_all

This function clears the multicast entry in the MRIB/MFIB.

This function is used to implement the `clear ipv6 mroute *` command.

Syntax

```
int
mrib6_api_clear_mroute_all (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_clear_mroute_g

This function clears the multicast route entries, which has the specified group value in the MRIB/MFIB.

This function is used to implement the `clear ipv6 mroute X:X::X:X` command.

Syntax

```
int
mrib6_api_clear_mroute_g (u_int32_t vr_id, vrf_id_t vrf_id,
                          struct pal_in6_addr *grp);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`grp`	Group IP address

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_INVALID_GROUP_ADDRESS when an invalid group IP address is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_clear_mroute_sg

This function clears the multicast route entry, which matches the specified source (unicast) address and the group (multicast) address (S,G) entry.

This function is used to implement the `clear ipv6 mroute X:X::X:X X:X::X:X` command.

Syntax

```
int
mrib6_api_clear_mroute_sg (u_int32_t vr_id, vrf_id_t vrf_id,
                           struct pal_in6_addr *src, struct pal_in6_addr *grp);
```

iInput Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`src`	Source IP address
`grp`	Group IP address

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_INVALID_GROUP_ADDRESS when an invalid group IP address is used

MRIB_API_SET_ERR_INVALID_SOURCE_ADDRESS when an invalid source address is used

MRIB_API_SET_ERR_MRT_NOT_EXIST when the IP multicast route does not exists

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_clear_mroute_stats_all

This function clears the IPv6 multicast statistics from the MRIB/MFIB.

This function is used to implement the `clear ipv6 mroute statistics *` command.

Syntax

```
int
mrib6_api_clear_mroute_stats_all (u_int32_t vr_id, vrf_id_t vrf_id);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

None

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_clear_mroute_stats_g

This function clears the multicast statics, which matches to the specific group entry.

This function is used to implement the `clear ipv6 mroute statistics X:X::X:X` command.

Syntax

```
int
mrib6_api_clear_mroute_stats_g (u_int32_t vr_id, vrf_id_t vrf_id,
                                struct pal_in6_addr *grp);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`grp`	Group IP address

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_INVALID_GROUP_ADDRESS when an invalid group IP address is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_clear_mroute_stats_sg

This function clears the multicast statistics, which matches the specified (S, G) entry.

This function is used to implement the `clear ipv6 mroute statistics X:X::X:X X:X::X:X` command.

Syntax

```
int
mrib6_api_clear_mroute_stats_sg (u_int32_t vr_id, vrf_id_t vrf_id,
                                 struct pal_in6_addr *src,
                                 struct pal_in6_addr *grp);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`src`	Source IP address
`grp`	Group IP address

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_ERR_INVALID_GROUP_ADDRESS when an invalid group IP address is used

MRIB_API_SET_ERR_INVALID_SOURCE_ADDRESS when an invalid source address is used

MRIB_API_SET_ERR_MRT_NOT_EXIST when the IP multicast route does not exists

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_all_set

This function enables debugging for all MRIBd events.

This function is used to implement the `debug ipv6 mrib all` command.

Syntax

```
int
mrib6_api_debug_all_set (const u_int32_t vr_id, const vrf_id_t vrf_id,
                         const int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_all_unset

This function disables debugging for all MRIBd events.

This function is used to implement the `no debug ipv6 mrib all` command.

Syntax

```
int
mrib6_api_debug_all_unset (const u_int32_t vr_id, const vrf_id_t vrf_id,
                           const int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_event_set

This function enables debugging of MRIB events.

This function is used to implement the `debug ipv6 mrib event` command.

Syntax

```
int
mrib6_api_debug_event_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_event_unset

This function disables debugging of MRIB events.

This function is used to implement the `no debug ipv6 mrib event` command.

Syntax

```
int
mrib6_api_debug_event_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_vif_set

This function enables MRIB virtual interface structure (VIF) debugging.

This function is used to implement the `debug ipv6 mrib vif` command.

Syntax

```
int
mrib6_api_debug_vif_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_vif_unset

This function disables MRIB VIF debugging.

This function is used to implement the `no debug ipv6 mrib vif` command.

Syntax

```
int
mrib6_api_debug_vif_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_mrt_set

This function enables MRIB Multicast Route Table (MRT) debugging.

This function is used to implement the `debug ipv6 mrib mrt` command.

Syntax

```
int
mrib6_api_debug_mrt_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_mrt_unset

This function disables MRIB MRT debugging.

This function is used to implement the `no debug ipv6 mrib mrt` command.

Syntax

```
int
mrib6_api_debug_mrt_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_stats_set

This function enables MRIB statistics debugging.

This function is used to implement the `debug ipv6 mrib stats` command.

Syntax

```
int
mrib6_api_debug_stats_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_stats_unset

This function disables MRIB statistics debugging.

This function is used to implement the `no debug ipv6 mrib stats` command in the exec and configure modes.

Syntax

```
int
mrib6_api_debug_stats_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_fib_msg_set

This function enables MRIB forwarding information base (FIB) message debugging.

This function is used to implement the `debug ipv6 mrib fib-msg` command.

Syntax

```
int
mrib6_api_debug_fib_msg_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_fib_msg_unset

This function disables MRIB FIB message debugging.

This function is used to implement the `no debug ipv6 mrib fib-msg` command in the exec and configure modes.

Syntax

```
int
mrib6_api_debug_fib_msg_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_register_msg_set

This function enables MRIB register message debugging.

This function is used to implement the `debug ipv6 mrib register-msg` command in the exec and configure modes.

Syntax

```
int
mrib6_api_debug_register_msg_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_register_msg_unset

This function disables MRIB register message debugging.

This function is used to implement the `no debug ipv6 mrib register-msg` command in the exec and configure modes.

Syntax

```
int
mrib6_api_debug_register_msg_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_nsm_msg_set

This function enables MRIB NSM message debugging.

This function is used to implement the `debug ipv6 mrib nsm-msg` command in the exec and configure modes.

Syntax

```
int
mrib6_api_debug_nsm_msg_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_nsm_msg_unset

This function disables MRIB NSM message debugging.

This function is used to implement the `no debug ipv6 mrib nsm-msg` command in the exec and configure modes.

Syntax

```
int
mrib6_api_debug_nsm_msg_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_mrib_msg_set

This function enables MRIB MRIB message debugging.

This function is used to implement the `debug ipv6 mrib mrib-msg` command in the exec and configure modes.

Syntax

```
int
mrib6_api_debug_mrib_msg_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_mrib_msg_unset

This function disables MRIB MRIB message debugging.

This function is used to implement the `no debug ipv6 mrib mrib-msg` command.

Syntax

```
int
mrib6_api_debug_mrib_msg_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_mtrace_set

This function enables MRIB mtrace debugging.

This function is used to implement the `debug ipv6 mrib mtrace` command.

Syntax

```
int
mrib6_api_debug_mtrace_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_mtrace_unset

This function disables MRIB mtrace debugging.

This function is used to implement the `no debug ipv6 mrib mtrace` command.

Syntax

```
int
mrib6_api_debug_mtrace_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_mtrace_detail_set

This function enables MRIB mtrace detailed debugging.

This function is used to implement the `debug ipv6 mrib mtrace-detail` command.

Syntax

```
int
mrib6_api_debug_mtrace_detail_set (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib6_api_debug_mtrace_detail_unset

This function disables MRIB mtrace detailed debugging.

This function is used to implement the `no debug ipv6 mrib mtrace-detail` command.

Syntax

```
int
mrib6_api_debug_mtrace_detail_unset (u_int32_t vr_id, vrf_id_t vrf_id, int cli_mode);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`cli_mode`	CLI command mode

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

CHAPTER 7 Multicast Routing MIB API

This chapter contains the Management Information Base (MIB) table API functions for the Multicast Routing Information Base (MRIB). These functions provide functionality for both IPv4 and IPv6 multicast network management.

Tables

ZebOS-XP supports Multicast Routing Management Information Base (MIB) tables described in draft-ietf-magma-mgmd-08.txt, the Multicast Group Membership Discovery MIB.

IP Multicast Route Table

This table contains multicast routing information for IP datagrams sent by particular sources to the IP multicast groups known to a router.

IP Multicast Routing Next Hop Table

This table contains information on next hops for IP multicast datagrams. Each entry is one of a list of next hops on outgoing interfaces for particular sources sending to a particular multicast group address.

IP Multicast Routing Interface Table

This table contains multicast routing information specific to interfaces.

Tables Not Supported

ZebOS-XP does not support the following tables:

- IP Multicast Host Interface Table
- IP Multicast Host Cache Table
- IP Multicast Reverse Host Cache Table
- IP Multicast Host Source List Table

API

Include File

These functions are in the `mribd/mrib_snmp_api.c` file and are declared in the `mribd/mrib_snmp_api.h` file.

mrib_snmp_api_get_mcast_enable

This function gets the setting that specifies whether multicast routing is enabled on a specified interface.

Syntax

```
int
mrib_snmp_api_get_mcast_enable (u_int32_t vr_id, vrf_id_t vrf_id, u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`val`	Pointer to the integer, which gets the current multicast routing state

Return Values

MRIB_API_SET_ERR_WRONG_VR when an invalid virtual router ID is used

MRIB_API_SET_ERR_WRONG_VRF when an invalid ID of a VPN routing/forwarding instance is used

MRIB_API_SET_SUCCESS when the function succeeds

mrib_snmp_api_set_mcast_enable

This function enables or disables multicast routing.

Syntax

```
int
mrib_snmp_api_set_mcast_enable (u_int32_t vr_id, vrf_id_t vrf_id, int val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`val`	The boolean value to enable and disable the multicast routing

Return Values

MRIB_API_SET_SUCCESS when the function succeeds

MRIB_API_SET_ERROR when the function fails

mrib_snmp_api_get_mcast_route_entry_count

This function gets the current route entry count.

Syntax

```
int
mrib_snmp_api_get_mcast_route_entry_count (u_int32_t vr_id, vrf_id_t vrf_id,
                                           u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`val`	Pointer to the integer, which gets the current route entry count

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_vif_ttl

This function gets the datagram TTL (Time To Live) threshold for the interface. Any IP multicast datagrams with a TTL less than this threshold will not be forwarded from the interface. The default value of 1 means all multicast packets are forwarded from the interface.

Syntax

```
int
mrib_snmp_api_get_vif_ttl (const u_int32_t vr_id, const vrf_id_t vrf_id,
                           struct mrib_snmp_api_vif_index *index,
                           u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The TTL value

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_vif_ttl

This function gets the datagram TTL threshold for the next interface. Any IP multicast datagrams with a TTL less than this threshold will not be forwarded from the interface. The default value of 1 means all multicast packets are forwarded from the interface.

Syntax

```
int
mrib_snmp_api_get_vif_ttl (const u_int32_t vr_id, const vrf_id_t vrf_id,
                           struct mrib_snmp_api_vif_index *index,
                           u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The TTL value

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_vif_ratelimit

This function gets the rate limit, in kilobits per second, of forwarded multicast traffic on the interface.

Syntax

```
int
mrib_snmp_api_get_vif_ratelimit (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                 struct mrib_snmp_api_vif_index *index,
                                 u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The rate-limit of forwarded multicast traffic on this interface.

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_vif_ratelimit

This function gets the rate limit, in kilobits per second, of forwarded multicast traffic on the next interface.

Syntax

```
int
mrib_snmp_api_get_next_vif_ratelimit (const u_int32_t vr_id,
                                      const vrf_id_t vrf_id,
                                      struct mrib_snmp_api_vif_index *index,
                                      u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The rate-limit of forwarded multicast traffic on the next interface.

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_set_vif_ratelimit

This function sets the rate limit, in kilobits per second, of forwarded multicast traffic on the interface.

Syntax

```
int
mrib_snmp_api_set_vif_ratelimit (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                 struct mrib_snmp_api_vif_index index, int val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID
`index`	Index to the entry in the multicast next hop table
`val`	The rate-limit value of forwarded multicast traffic on this interface.

Output Parameters

None

Return Values

MRIB_API_SET_ERROR when the function fails. Rate limit is not supported.

mrib_snmp_api_get_vif_inmcastoctets

This function gets the number of octets of multicast packets that have arrived on the interface.

Syntax

```
int
mrib_snmp_api_get_vif_inmcastoctets (const u_int32_t vr_id, const vrf_id_t vrf_id,
struct mrib_snmp_api_vif_index *index, ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets of received multicast packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_vif_inmcastoctets

This function gets the number of octets of multicast packets that have arrived on the next interface.

Syntax

```
int
mrib_snmp_api_get_next_vif_inmcastoctets (const u_int32_t vr_id,
                                          const vrf_id_t vrf_id,
                                          struct mrib_snmp_api_vif_index *index,
                                          ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets of received multicast packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_vif_outmcastoctets

This function gets the number of octets of multicast packets that have been sent on the interface.

Syntax

```
int
mrib_snmp_api_get_vif_outmcastoctets (const u_int32_t vr_id, const vrf_id_t vrf_id,
struct mrib_snmp_api_vif_index *index, ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets of sent multicast packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_vif_outmcastoctets

This function gets the number of octets of multicast packets that have been sent on the next interface.

Syntax

```
int
mrib_snmp_api_get_next_vif_outmcastoctets (const u_int32_t vr_id,
                                           const vrf_id_t vrf_id,
                                           struct mrib_snmp_api_vif_index *index,
                                           ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets of sent multicast packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_vif_inmcastpkts

This function gets the number of multicast packets that have arrived on the interface.

Syntax

```
int
mrib_snmp_api_get_vif_inmcastpkts (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                   struct mrib_snmp_api_vif_index *index,
                                   ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of multicast packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_vif_inmcastpkts

This function gets the number of octets of multicast packets that have arrived on the next interface.

Syntax

```
int
mrib_snmp_api_get_next_vif_inmcastpkts (const u_int32_t vr_id,
                                        const vrf_id_t vrf_id,
                                        struct mrib_snmp_api_vif_index *index,
                                        ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of multicast packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_vif_outmcastpkts

This function gets the number of multicast packets that have been sent by this interface.

Syntax

```
int
mrib_snmp_api_get_vif_outmcastpkts (const u_int32_t vr_id,
                                    const vrf_id_t vrf_id,
                                    struct mrib_snmp_api_vif_index *index,
                                    ut_int64_t *val);
```

Input Parameters

vr_id	Virtual router ID
vrf_id	VPN routing/forwarding instance ID

Output Parameters

index	Index to the entry in the multicast next hop table
val	The number of multicast packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_vif_outmcastpkts

This function gets the number of multicast packets that have been sent by this interface.

Syntax

```
int
mrib_snmp_api_get_next_vif_outmcastpkts (const u_int32_t vr_id,
                                         const vrf_id_t vrf_id,
                                         struct mrib_snmp_api_vif_index *index,
                                         ut_int64_t *val);
```

Input Parameters

vr_id	Virtual router ID
vrf_id	VPN routing/forwarding instance ID

Output Parameters

index	Index to the entry in the multicast next hop table
val	The number of multicast packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_vif_discontinuity_time

This function retrieves the last time a discontinuity occurred in any of the counters associated with this interface.

Syntax

```
int
mrib_snmp_api_get_vif_discontinuity_time (const u_int32_t vr_id,
                                          const vrf_id_t vrf_id,
                                          struct mrib_snmp_api_vif_index *index,
                                          u_int32_t  *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	Pointer to the integer, which gets the current discontinuity time.

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_vif_discontinuity_time

This function retrieves the last time a discontinuity occurred in any of the counters associated with the next interface.

Syntax

```
int
mrib_snmp_api_get_next_vif_discontinuity_time (const u_int32_t vr_id,
                                               const vrf_id_t vrf_id,
                                               struct mrib_snmp_api_vif_index *index,
                                               u_int32_t  *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	Pointer to the integer, which gets the current discontinuity time.

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_upstreamneighbor_type

This function gets the address type of the upstream neighbor (RPF neighbor) from which IP datagrams from the source to this multicast address are received, or 0.0.0.0 if the address is unknown.

Syntax

```
int
mrib_snmp_api_get_mrt_upstreamneighbor_type (const u_int32_t vr_id,
                                             const vrf_id_t vrf_id,
                                             struct mrib_snmp_api_mrt_index *index,
                                             u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The IP address of the upstream neighbor

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_upstreamneighbor_type

This function gets the address type of the next upstream neighbor from which IP datagrams from the source to this multicast address are received, or 0.0.0.0 if the address is unknown.

Syntax

```
int
mrib_snmp_api_get_mrt_upstreamneighbor_type (const u_int32_t vr_id,
                                             const vrf_id_t vrf_id,
                                             struct mrib_snmp_api_mrt_index *index,
                                             u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The IP address of the upstream neighbor

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_upstreamneighbor

This function gets the address of the upstream neighbor from which IP datagrams from the source to this multicast address are received, or 0.0.0.0 if the upstream neighbor is unknown.

Syntax

```
int
mrib_snmp_api_get_mrt_upstreamneighbor (const u_int32_t vr_id,
                                        const vrf_id_t vrf_id,
                                        struct mrib_snmp_api_mrt_index *index,
                                        struct prefix *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The IP address of the upstream neighbor

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_upstreamneighbor

This function gets the address of the upstream neighbor which IP datagrams from the next source to the multicast address are received, or 0.0.0.0 if the upstream neighbor is unknown.

Syntax

```
int
mrib_snmp_api_get_next_mrt_upstreamneighbor (const u_int32_t vr_id,
                                             const vrf_id_t vrf_id,
                                             struct mrib_snmp_api_mrt_index *index,
                                             struct prefix *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The IP address of the upstream neighbor

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_inifindex

This function gets the value of ifIndex (interface index) for the interface on which IP datagrams sent by the source to this multicast address are received.

Syntax

```
int
mrib_snmp_api_get_mrt_inifindex (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                 struct mrib_snmp_api_mrt_index *index,
                                 u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The value of ifIndex (interface index)

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_inifindex

This function gets the value of ifIndex (interface index) for the next interface on which IP datagrams sent by the source to this multicast address are received.

Syntax

```
int
mrib_snmp_api_get_next_mrt_inifindex (const u_int32_t vr_id,
                                      const vrf_id_t vrf_id,
                                      struct mrib_snmp_api_mrt_index *index,
                                      u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The value of ifIndex (interface index)

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_timestamp

This function gets the timestamp of the interface on which IP datagrams sent by the source to this multicast address are received

Syntax

```
int
mrib_snmp_api_get_mrt_timestamp (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                 struct mrib_snmp_api_mrt_index *index,
                                 u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The interface timestamp

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_timestamp

This function gets the timestamp of the next interface on which IP datagrams sent by the source to this multicast address are received.

Syntax

```
int
mrib_snmp_api_get_next_mrt_timestamp (const u_int32_t vr_id,
                                      const vrf_id_t vrf_id,
                                      struct mrib_snmp_api_mrt_index *index,
                                      u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The interface timestamp

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_expirytime

This function gets the minimum amount of time remaining before this entry is aged out. The value 0 indicates that the entry is not subject to aging.

Syntax

```
int
mrib_snmp_api_get_mrt_expirytime (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                  struct mrib_snmp_api_mrt_index *index,
                                  u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The expiration time

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_expirytime

This function gets the minimum amount of time remaining before the next entry is aged out. The value 0 indicates that the entry is not subject to aging.

Syntax

```
int
mrib_snmp_api_get_next_mrt_expirytime (const u_int32_t vr_id,
                                       const vrf_id_t vrf_id,
                                       struct mrib_snmp_api_mrt_index *index,
                                       u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The expiration time

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_protocol

This function gets the multicast routing protocol by which this multicast forwarding entry was learned.

Syntax

```
int
mrib_snmp_api_get_mrt_protocol (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                struct mrib_snmp_api_mrt_index *index,
                                u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The protocol ID defined as IANAipMRouteProtocol

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_protocol

This function gets the multicast routing protocol by which the next multicast forwarding entry was learned.

Syntax

```
int
mrib_snmp_api_get_next_mrt_protocol (const u_int32_t vr_id,
                                     const vrf_id_t vrf_id,
                                     struct mrib_snmp_api_mrt_index *index,
                                     u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The protocol ID defined as IANAipMRouteProtocol

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_rtprotocol

This function gets the routing mechanism by which the route used to find the upstream or parent interface for this multicast forwarding entry was learned.

Syntax

```
int
mrib_snmp_api_get_mrt_rtprotocol (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                  struct mrib_snmp_api_mrt_index *index,
                                  u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The protocol ID defined as IANAipMRouteProtocol

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_rtprotocol

This function gets the routing mechanism by which the route used to find the upstream or parent interface for the next multicast forwarding entry was learned.

Syntax

```
int
mrib_snmp_api_get_next_mrt_rtprotocol (const u_int32_t vr_id,
                                       const vrf_id_t vrf_id,
                                       struct mrib_snmp_api_mrt_index *index,
                                       u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The protocol ID defined as IANAipMRouteProtocol

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_rtaddr_type

This function gets the address type of the address portion of the route used to find the upstream or parent interface for this multicast entry.

Syntax

```
int
mrib_snmp_api_get_mrt_rtaddr_type (const u_int32_t vr_id,
                                   const vrf_id_t vrf_id,
                                   struct mrib_snmp_api_mrt_index *index,
                                   u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The address type

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_rtaddr_type

This function gets the address type of the address portion of the route used to find the next upstream or parent interface for this multicast entry.

Syntax

```
int
mrib_snmp_api_get_next_mrt_rtaddr_type (const u_int32_t vr_id,
                                        const vrf_id_t vrf_id,
                                        struct mrib_snmp_api_mrt_index *index,
                                        u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The address type

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_rtaddr

This function gets the address portion of the route used to find the upstream or parent interface for this multicast forwarding entry.

Syntax

```
int
mrib_snmp_api_get_mrt_rtaddr (const u_int32_t vr_id, const vrf_id_t vrf_id,
                              struct mrib_snmp_api_mrt_index *index,
                              struct prefix *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The IP address

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_rtaddr

This function gets the address portion of the route used to find the upstream or parent interface for the next multicast forwarding entry.

Syntax

```
int
mrib_snmp_api_get_next_mrt_rtaddr (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                   struct mrib_snmp_api_mrt_index *index,
                                   struct prefix *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The IP address

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_rtprefix_len

This function gets the prefix length (an integer) of the route used to find the upstream or parent interface for this multicast forwarding entry.

Syntax

```
int
mrib_snmp_api_get_mrt_rtprefix_len (const u_int32_t vr_id,
                                    const vrf_id_t vrf_id,
                                    struct mrib_snmp_api_mrt_index *index,
                                    u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The prefix length of the route

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_rtprefix_len

This function gets the prefix length (an integer) of the route used to find the upstream or parent interface for the next multicast forwarding entry.

Syntax

```
int
mrib_snmp_api_get_next_mrt_rtprefix_len (const u_int32_t vr_id,
                                         const vrf_id_t vrf_id,
                                         struct mrib_snmp_api_mrt_index *index,
                                         u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The prefix length of the route

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_rttype

This function gets the reason the given route was placed in the (logical) MRIB. A value of unicast means that the route would normally be placed only in the unicast RIB, but was placed in the MRIB (instead or in addition) due to local configuration, such as when running PIM over RIP. A value of multicast means that the route was explicitly added to the MRIB by the routing protocol, such as Static Multicast Routing.

Syntax

```
int
mrib_snmp_api_get_mrt_rttype (const u_int32_t vr_id, const vrf_id_t vrf_id,
                              struct mrib_snmp_api_mrt_index *index,
                              u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The reason: Unicast is 1, Multicast is 2

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_rttype

This function gets the reason the next given route was placed in the (logical) MRIB. A value of unicast means that the route would normally be placed only in the unicast RIB, but was placed in the MRIB (instead or in addition) due to local configuration, such as when running PIM over RIP. A value of multicast means that the route was explicitly added to the MRIB by the routing protocol, such as Static Multicast Routing.

Syntax

```
int
mrib_snmp_api_get_next_mrt_rttype (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                   struct mrib_snmp_api_mrt_index *index,
                                   u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The reason: Unicast is 1, Multicast is 2

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_octets

This function gets the number of octets contained in IP datagrams received from the source and addressed to this multicast group address, which were forwarded by this router.

Syntax

```
int
mrib_snmp_api_get_mrt_octets (const u_int32_t vr_id, const vrf_id_t vrf_id,
                              struct mrib_snmp_api_mrt_index *index,
                              ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_octets

This function gets the number of octets contained in IP datagrams received from the next source and multicast group address, which were forwarded by this router.

Syntax

```
int
mrib_snmp_api_get_next_mrt_octets (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                   struct mrib_snmp_api_mrt_index *index,
                                   ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_pkts

This function gets the number of packets of IP datagrams received from the source and addressed to this multicast group address that were forwarded by this router.

Syntax

```
int
mrib_snmp_api_get_mrt_pkts (const u_int32_t vr_id, const vrf_id_t vrf_id,
                            struct mrib_snmp_api_mrt_index *index,
                            ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_pkts

This function gets the number of packets of IP datagrams received from the source and addressed to the next multicast group address that were forwarded by this router.

Syntax

```
int
mrib_snmp_api_get_next_mrt_pkts (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                 struct mrib_snmp_api_mrt_index *index,
                                 ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_mrt_diffinifoctets

This function gets the number of octets of IP datagrams this router received from the source and addressed to this multicast group address that were dropped because they were not received on the interface indicated by ipMRouteInIfIndex.

Syntax

```
int
mrib_snmp_api_get_mrt_diffinifoctets (const u_int32_t vr_id,
                                      const vrf_id_t vrf_id,
                                      struct mrib_snmp_api_mrt_index *index,
                                      ut_int64_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_mrt_diffinifoctets

This function gets the number of octets of IP datagrams this router received from the source and addressed to the next multicast group address that were dropped because they were not received on the interface indicated by ipMRouteInIfIndex.

Syntax

```
int
mrib_snmp_api_get_next_mrt_diffinifoctets (const u_int32_t vr_id,
                                           const vrf_id_t vrf_id,
                                           struct mrib_snmp_api_mrt_index *index,
                                           ut_int64_t  *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_nh_state

This function gets the indication of whether the outgoing interface and next-hop represented by this entry is currently being used to forward IP datagrams. The value 'forwarding' indicates it is currently being used; the value 'pruned' indicates it is not.

Syntax

```
int
mrib_snmp_api_get_nh_state (const u_int32_t vr_id, const vrf_id_t vrf_id,
                            struct mrib_snmp_api_nh_index *index,
                            u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The state of nexthop

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_nh_state

This function gets the indication of whether the outgoing interface and next-hop represented by the next entry is currently being used to forward IP datagrams. The value 'forwarding' indicates it is currently being used; the value 'pruned' indicates it is not.

Syntax

```
int
mrib_snmp_api_get_next_nh_state (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                 struct mrib_snmp_api_nh_index *index,
                                 u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The state of nexthop

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_nh_uptime

This function gets the time since the multicast routing information represented by this entry was learned by the router.

Syntax

```
int
mrib_snmp_api_get_nh_uptime (const u_int32_t vr_id, const vrf_id_t vrf_id,
                             struct mrib_snmp_api_nh_index *index,
                             u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The uptime

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_nh_uptime

This function gets the time since the multicast routing information represented by the next entry was learned by the router.

Syntax

```
int
mrib_snmp_api_get_next_nh_uptime (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                  struct mrib_snmp_api_nh_index *index,
                                  u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The uptime

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_nh_expirytime

This function gets the minimum amount of time remaining before the next entry will be aged out. If the value of ipMcastrouteNextHopState is pruned (1), it is the remaining time until the prune expires and the state reverts to forwarding (2). Otherwise, it is the remaining time until this entry is removed from the table. The time remaining may be copied from ipMcastrouteExpiryTime, if the protocol in use for this entry does not specify next-hop timers. A value of 0 indicates that the entry is not subject to aging.This function gets the minimum amount of time remaining before the next entry is aged out.

Syntax

```
int
mrib_snmp_api_get_nh_expirytime (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                 struct mrib_snmp_api_nh_index *index,
                                 u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The expiration time

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_nh_expirytime

This function gets the minimum amount of time remaining before this entry is aged out. If ipMcastrouteNextHopState is pruned (1), it is the remaining time until the prune expires and the state reverts to forwarding (2). Otherwise, it is the remaining time until this entry is removed from the table. The time remaining may be copied from ipMcastrouteExpiryTime if the protocol in use for this entry does not specify next-hop timers. A value of 0 indicates that the entry is not subject to aging.

Syntax

```
int
mrib_snmp_api_get_next_nh_expirytime (const u_int32_t vr_id,
                                      const vrf_id_t vrf_id,
                                      struct mrib_snmp_api_nh_index *index,
                                      u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The expiration time

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_nh_closestmemberhops

This function gets the minimum number of hops between this router and any member of this IP multicast group reached through this next-hop on this outgoing interface. Any IP multicast datagram for the group which has a TTL (Time To Live) less than this number of hops is not forwarded to this next-hop.

Syntax

```
int
mrib_snmp_api_get_nh_closestmemberhops (const u_int32_t vr_id,
                                        const vrf_id_t vrf_id,
                                        struct mrib_snmp_api_nh_index *index,
                                        u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The closest member hops

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_nh_closestmemberhops

This function gets the minimum number of hops between this router and any member of this IP multicast group reached through this next-hop on the next outgoing interface. Any IP multicast datagram for the group which has a TTL less than this number of hops is forwarded to this next-hop.

Syntax

```
int
mrib_snmp_api_get_next_nh_closestmemberhops (const u_int32_t vr_id,
                                             const vrf_id_t vrf_id,
                                             struct mrib_snmp_api_nh_index *index,
                                             u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The closest member hops

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_nh_protocol

This function gets the multicast routing protocol by which this multicast forwarding entry was learned.

Syntax

```
int
mrib_snmp_api_get_nh_protocol (const u_int32_t vr_id, const vrf_id_t vrf_id,
                               struct mrib_snmp_api_nh_index *index,
                               u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The protocol ID defined as IANAipMRouteProtocol

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_nh_protocol

This function gets the multicast routing protocol by which the next multicast forwarding entry was learned.

Syntax

```
int
mrib_snmp_api_get_next_nh_protocol (const u_int32_t vr_id,
                                    const vrf_id_t vrf_id,
                                    struct mrib_snmp_api_nh_index *index,
                                    u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The protocol ID defined as IANAipMRouteProtocol

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_nh_octets

This function gets the number of octets contained in IP datagrams received from the source and addressed to this multicast group address, which were forwarded by this router.

Syntax

```
int
mrib_snmp_api_get_nh_octets (const u_int32_t vr_id, const vrf_id_t vrf_id,
                             struct mrib_snmp_api_nh_index *index,
                             u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_nh_octets

This function gets the number of octets contained in IP datagrams received from the next source and multicast group address, which were forwarded by this router.

Syntax

```
int
mrib_snmp_api_get_next_nh_octets (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                  struct mrib_snmp_api_nh_index *index,
                                  u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of octets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_nh_pkts

This function gets the number of packets that have been forwarded using this route.

Syntax

```
int
mrib_snmp_api_get_nh_pkts (const u_int32_t vr_id, const vrf_id_t vrf_id,
                           struct mrib_snmp_api_nh_index *index,
                           u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

mrib_snmp_api_get_next_nh_pkts

This function gets the number of packets which have been forwarded using the next route.

Syntax

```
int
mrib_snmp_api_get_next_nh_pkts (const u_int32_t vr_id, const vrf_id_t vrf_id,
                                struct mrib_snmp_api_nh_index *index,
                                u_int32_t *val);
```

Input Parameters

`vr_id`	Virtual router ID
`vrf_id`	VPN routing/forwarding instance ID

Output Parameters

`index`	Index to the entry in the multicast next hop table
`val`	The number of packets

Return Values

MRIB_API_GET_SUCCESS when the function succeeds

MRIB_API_GET_ERROR when the function fails

CHAPTER 8 MRIBv4 IGMP Command API

This chapter describes the command API for the Multicast Routing Information Base IPv4 (MRIBv4) Internet Group Management Protocol (IGMP).

Include File

To call the functions in this chapter, you must include `mribd/mrib4/igmp/mrib4_igmp_api.h`.

Instance-Level Configuration API

The functions in this section enable instance-level commands.

mrib4_igmp_api_limit_set

This function sets the limit for group-record states across all interfaces in the specified IGMP Instance. An exception access-list can be specified to exclude certain groups from being subject to this limit value.

Syntax

```
int
mrib4_igmp_api_limit_set (struct igmp_instance *igi,
                          u_int32_t limit,
                          u_int8_t *except_alist)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`limit`	Limit value
`except_alist`	Pointer to an access-list name

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_OOM when a memory allocation error occurs

mrib4_igmp_api_limit_unset

This function unsets the limit and exception access-list for group-record states across the specified IGMP instance.

Syntax

```
int
mrib4_igmp_api_limit_unset (struct igmp_instance *igi)
```

Input Parameters

`igi`	Pointer to the IGMP instance

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

mrib4_igmp_api_ssm_map_enable_set

This function sets the IGMP SSM-Map enable at the instance level.

Syntax

```
int
mrib4_igmp_api_ssm_map_enable_set (struct igmp_instance *igi);
```

Input Parameters

`igi`	Pointer to the IGMP instance

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

mrib4_igmp_api_ssm_map_enable_unset

This function unsets IGMP SSM-Map enable at instance level.

Syntax

```
int
mrib4_igmp_api_ssm_map_enable_unset (struct igmp_instance *igi);
```

Input Parameters

`igi`	Pointer to the IGMP instance

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

mrib4_igmp_api_ssm_map_static_set

This function sets an SSM map definition. The supplied source-address string will be used to produce the (G, S) SSM mapping for the group address defined as the supplied access-list reference string. This function may be invoked multiple times to define multiple SSM mappings.

Syntax

```
int
mrib4_igmp_api_ssm_map_static_set (struct igmp_instance *igi,
                                   u_int8_t *alist,
                                   u_int8_t *msrc_arg)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`alist`	Pointer to the group(s) address access-list name
`msrc_arg`	Pointer to the source-address string

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_OOM when a memory allocation error occurs

mrib4_igmp_api_ssm_map_static_unset

This function unsets the IGMP SSM-Map Static at the instance level.

This function unsets the SSM map definition identified by the supplied access-list reference string and source-address string.

Syntax

```
int
mrib4_igmp_api_ssm_map_static_unset (struct igmp_instance *igi,
                                     u_int8_t *alist,
                                     u_int8_t *msrc_arg);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`alist`	Pointer to the group(s) address access-list name
`msrc_arg`	Pointer to the source-address string

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_VALUE when no such configured value is found

mrib4_igmp_api_clear

This function clears IGMP state information.

Syntax

```
int
mrib4_igmp_api_clear (struct igmp_instance *igi,
                      struct interface *ifp,
                      struct pal_in4_addr *pgrp,
                      struct pal_in4_addr *psrc);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the specific interface to clear IGMP information. If NULL, IGMP group and source records on all interfaces belonging to the IGMP instance are cleared.
`pgrp`	Pointer to the specific IGMP group address to be cleared. If NULL, all IGMP group and source records are cleared.
`psrc`	Pointer to the specific IGMP source address to be cleared. If NULL, all IGMP source records are cleared. Cannot be non-NULL when ifp or pgrp are NULL.

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_MALFORMED_ARG when the argument is malformed

Interface-Level Configuration API

The functions in this section enable interface-level configuration commands.

mrib4_igmp_api_if_set

This function enables IGMP on an interface.

Syntax

```
int
mrib4_igmp_api_if_set (struct igmp_instance *igi,
                       struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

RESULT_OK when the function succeeds

mrib4_igmp_api_if_unset

This function disables IGMP on an interface.

Syntax

```
int
mrib4_igmp_api_if_unset (struct igmp_instance *igi,
                         struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_access_list_set

This function sets the IGMP access list.

Syntax

```
int
mrib4_igmp_api_if_access_list_set (struct igmp_instance *igi,
                                   struct interface *ifp,
                                   u_int8_t *alist)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`alist`	Pointer to the access-list name

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_access_list_unset

This function unsets the IGMP access list.

Syntax

```
int
mrib4_igmp_api_if_access_list_unset (struct igmp_instance *igi,
                                     struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_immediate_leave_set

This function sets the IGMP immediate-leave access-list.

Syntax

```
int
mrib4_igmp_api_if_immediate_leave_set (struct igmp_instance *igi,
                                       struct interface *ifp,
                                       u_int8_t *alist)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`alist`	Pointer to the access-list name

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_immediate_leave_unset

This function unsets the IGMP immediate-leave access-list.

Syntax

```
int
mrib4_igmp_api_if_immediate_leave_unset (struct igmp_instance *igi,
                                         struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_lmqc_set

This function sets the last-member query-count (LMQC) value.

Syntax

```
int
mrib4_igmp_api_if_lmqc_set (struct igmp_instance *igi,
                            struct interface *ifp,
                            u_int32_t lmqc);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`lmqc`	LMQC value

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_lmqc_unset

This function unsets the last-member query-count (LMQC) value.

Syntax

```
int
mrib4_igmp_api_if_lmqc_unset (struct igmp_instance *igi,
                              struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_lmqi_set

This function sets the last-member query-interval (LMQI) value.

Syntax

```
int
mrib4_igmp_api_if_lmqi_set (struct igmp_instance *igi,
                            struct interface *ifp,
                            u_int32_t lmqi,
                            bool_t is_lmqi);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`is_lmqi`	LMQI value

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_CFG_FOR_PROXY_SERVICE when the interface is configured for proxy service; undo first

IGMP_ERR_NO_CFG_FOR_PROXY_SERVICE when a non proxy-service interface has an invalid configuration

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_lmqi_unset

This function unsets the last-member query-interval to the default value.

Syntax

```
int
mrib4_igmp_api_if_lmqi_unset (struct igmp_instance *igi,
                              struct interface *ifp,
                              bool_t is_lmqi);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`is_lmqi`	LMQI value

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_mroute_pxy_set

This function sets up the proxy-service interface association for the specified interface.

Syntax

```
int
mrib4_igmp_api_if_mroute_pxy_set (struct igmp_instance *igi,
                                  struct interface *ifp,
                                  u_int8_t *mrtr_pxy_ifname);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the downstream interface in the IGMP Proxy configuration
`mrtr_pxy_ifname`	Pointer to the interface name of the upstream proxy-service interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_CFG_FOR_PROXY_SERVICE when the interface is configured for proxy service; undo first

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_mroute_pxy_unset

This function unsets the proxy-service association on the specified downstream interface.

Syntax

```
int
mrib4_igmp_api_if_mroute_pxy_unset (struct igmp_instance *igi,
                                    struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the downstream interface in the IGMP Proxy configuration

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_pxy_service_set

This function sets the specified interface for IGMP proxy service.

Syntax

```
mrib4_igmp_api_if_pxy_service_set (struct igmp_instance *igi,
                                   struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface that provides upstream host-side IGMP Proxy-service

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_CFG_WITH_MROUTE_PROXY when the interface is configured with mroute proxy; undo first

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_pxy_service_unset

This function unsets IGMP Proxy service on the specified interface.

Syntax

```
int
mrib4_igmp_api_if_pxy_service_unset (struct igmp_instance *igi,
                                     struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface that provides upstream host-side IGMP Proxy-service

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_querier_timeout_set

This function sets the IGMP other-querier timeout.

Syntax

```
int
mrib4_igmp_api_if_querier_timeout_set (struct igmp_instance *igi,
                                       struct interface *ifp,
                                       u_int16_t other_querier_interval)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`other_querier_interval`	The value for IGMP other-querier timeout

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_querier_timeout_unset

This function unsets the IGMP other-querier timeout.

Syntax

```
int
mrib4_igmp_api_if_querier_timeout_unset (struct igmp_instance *igi,
                                         struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_query_interval_set

This function sets the IGMP query interval value.

Syntax

```
int
mrib4_igmp_api_if_query_interval_set (struct igmp_instance *igi,
                                      struct interface *ifp,
                                      u_int32_t query_interval)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`query_interval`	The IGMP query interval value

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_query_interval_unset

This function unsets the IGMP query interval value.

Syntax

```
int
mrib4_igmp_api_if_query_interval_unset (struct igmp_instance *igi,
                                        struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_query_response_interval_set

This function sets the IGMP query-response interval value.

Syntax

```
int
mrib4_igmp_api_if_query_response_interval_set (struct igmp_instance *igi,
                                               struct interface *ifp,
                                               u_int32_t response_interval)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`response_interval`	The IGMP query-response interval value

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_query_response_interval_unset

This function unsets the IGMP query-response interval value.

Syntax

```
int
mrib4_igmp_api_if_query_response_interval_unset (struct igmp_instance *igi,
                                                 struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_robustness_var_set

This function sets the robustness variable.

Syntax

```
int
mrib4_igmp_api_if_robustness_var_set (struct igmp_instance *igi,
                                      struct interface *ifp,
                                      u_int32_t robustness_var);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`robustness_var`	The value of the robustness variable

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_robustness_var_unset

This function unsets the robustness variable.

Syntax

```
int
mrib4_igmp_api_if_robustness_var_unset (struct igmp_instance *igi,
                                        struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_version_set

This function sets the IGMP version.

Syntax

```
int
mrib4_igmp_api_if_version_set (struct igmp_instance *igi,
                               struct interface *ifp,
                               u_int16_t version);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`version`	The IGMP version number

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_version_unset

This function unsets the IGMP version.

Syntax

```
int
mrib4_igmp_api_if_version_unset (struct igmp_instance *igi,
                                 struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_offlink_if_set

This function sets the IGMP offlink support on the specified interface.

Syntax

```
int
mrib4_igmp_api_offlink_if_set (struct igmp_instance *igi,
                               struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_offlink_if_unset

This function unsets the IGMP offlink support on the specified interface.

Syntax

```
int
mrib4_igmp_api_offlink_if_unset (struct igmp_instance *igi,
                                 struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_startup_query_interval_set

This function sets the IGMP startup query interval value.

Syntax

```
int
mrib4_igmp_api_if_startup_query_interval_set (struct igmp_instance *igi,
                                              struct interface *ifp,
                                              u_int32_t query_interval)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`query_interval`	The IGMP query interval value

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_startup_query_interval_unset

This function unsets the IGMP startup query interval value.

Syntax

```
int
mrib4_igmp_api_if_startup_query_interval_unset (struct igmp_instance *igi,
                                                struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_startup_query_count_set

This function sets the IGMP startup query count.

Syntax

```
int
mrib4_igmp_api_if_startup_query_count_set (struct igmp_instance *igi,
                                           struct interface *ifp,
                                           u_int32_t query_count)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`query_count`	The IGMP startup query count

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_startup_query_count_unset

This function unsets the IGMP startup query count.

Syntax

```
int
mrib4_igmp_api_if_startup_query_count_unset (struct igmp_instance *igi,
                                             struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_ra_set

This function sets the strict RA option validation.

Syntax

```
int
mrib4_igmp_api_if_ra_set (struct igmp_instance *igi,
                          struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_ra_unset

This function unsets the strict RA option validation.

Syntax

```
int
mrib4_igmp_api_if_ra_unset (struct igmp_instance *igi,
                            struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

IGMP_ERR_MALFORMED_ARG when the argument is malformed

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_NO_SUCH_SOURCE_REC when no such source record is found

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

IGMP_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

IGMP_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib4_igmp_api_if_static_join_group_source_set

This function sets the static (S, G) entry on the specified interface.

Syntax

```
int
mrib4_igmp_api_if_static_join_group_source_set (struct igmp_instance *igi,
                                                struct interface *ifp,
                                                struct pal_in4_addr *pgrp,
                                                struct pal_in4_addr *psrc,
                                                u_int8_t *ifname,
                                                bool_t is_ssm_mapped,
                                                bool_t is_local);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`pgrp`	Pointer to the specific IGMP group address
`psrc`	Pointer to the specific IGMP source address
`ifname`	The interface name
`is_ssm_mapped`	Whether the entry is SSM-mapped
`is_local`	Local flag of the IGMP group record

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

RESULT_OK when the function succeeds

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IIGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

mrib4_igmp_api_if_static_join_group_source_unset

This function unsets the IGMP static group.

Syntax

```
int
mrib4_igmp_api_if_static_join_group_source_unset (struct igmp_instance *igi,
                                                  struct interface *ifp,
                                                  struct pal_in4_addr *pgrp,
                                                  struct pal_in4_addr *psrc,
                                                  u_int8_t *ifname,
                                                  bool_t is_ssm_mapped,
                                                  bool_t is_local);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the interface
`pgrp`	Pointer to the specific IGMP group address
`psrc`	Pointer to the specific IGMP source address
`ifname`	The interface name
`is_ssm_mapped`	Whether the entry is SSM-mapped
`is_local`	Local flag of the IGMP group record

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

IGMP_ERR_MALFORMED_ARG when the argument is malformed

RESULT_OK when the function succeeds

IGMP_ERR_DOOM when the IGMP protocol fails

IGMP_ERR_BUF_TOO_SHORT when the processing exceeds buffer space

IGMP_ERR_NO_SRC_ENC when no such source is encountered

IIGMP_ERR_GENERIC when the operation fails

IGMP_ERR_NO_VALID_CONFIG when there is an invalid configuration for this operation

mrib4_igmp_api_static_group_source_flag_unset

This function unsets the IGMP static group.

Syntax

```
void
mrib4_igmp_api_static_group_source_flag_unset (struct igmp_group_rec *igr,
                                               bool_t is_local);
```

Input Parameters

`igr`	Pointer to the IGMP IGMP group record
`is_local`	Local flag of the IGMP group record

Output Parameters

None

Return Values

None

CHAPTER 9 L2 IGMP Snooping Command API

This chapter describes the L2 Internet Group Management Protocol (IGMP) snooping functions.

In ZebOS-XP, the L2MRIBd process maintains snooping-related functionality.

Overview

The L2MRIBd process primarily interacts with:

- NSM, for bridge, VLAN and port- related information and their association with each other
- IMI for snooping related CLI's
- MSTPd for spanning tree related information
- MRIBd to find out whether IGMP is enabled on any VLAN or not
- HAL for updating snooping entries in hardware

A common IGMP/MLD library is used for both IGMP/MLD snooping and IGMP/MLD multicast routing.To create the L2MRIBd process, either IGMP or MLD snooping is enabled in the configuration file. This configuration file enables the HAVE_L2MRIBD build flag.

How L2mribd Interacts with Protocols

Figure 9-1 shows how the logical architecture of L2MRIBd.h communicates with other protocols.

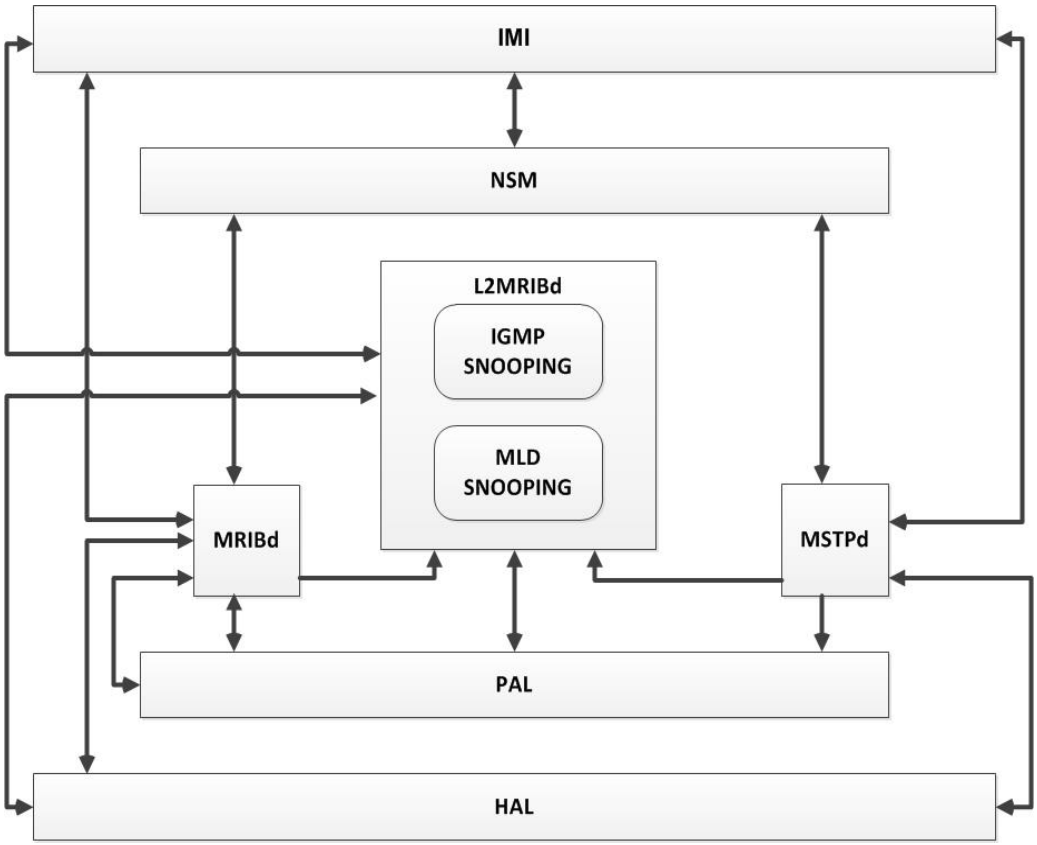

Figure 9-1: L2MRIB interacts with different protocols

- Initializes global libs and starts the protocol daemon.
- Starts NSM client to communicate with NSM.
- Creates the L2mcast, which holds the IGMP/MLD instances and associates input and output buffers.
- Starts the L2MRIB server to communicate with MSTPd.
- Starts the MRIB Client to communicate with MRIBd.
- Processes the IGMP/MLD join/leave/query messages.
- Updates the hardware with related snooping information.

Include File

The `igmp.h` file is the only one required to be included in `*.c` files outside of the `lib/igmp/` directory where IGMP-related functionality is referenced. The function declaration for the functions in this chapter is made available by including `igmp.h`.

Instance-Level Configuration API

The functions in this section enable instance-level commands.

igmp_snooping_set

This function enables IGMP Snooping on all interfaces of this instance not explicitly (individually) disabled for IGMP Snooping. IGMP Snooping is globally enabled by default.

API Call

```
s_int32_t
igmp_snooping_set (struct igmp_instance *igi)
```

Input Parameters

`igi`	Pointer to the IGMP instance

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

igmp_snooping_unset

This function disables IGMP Snooping on all interfaces of this instance not explicitly (individually) enabled for IGMP Snooping.

API Call

```
s_int32_t
igmp_snooping_unset (struct igmp_instance *igi)
```

Input Parameters

`igi`	Pointer to the IGMP instance

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

Interface-Level Configuration API

The functions in this section enable interface-level configuration commands.

igmp_if_snooping_set

This function explicitly enables IGMP Snooping on the specified VLAN interface.

API Call

```
s_int32_t
igmp_if_snooping_set (struct igmp_instance *igi,
                      struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

igmp_if_snooping_unset

This function explicitly disables IGMP Snooping on the specified VLAN interface. It also unsets all configuration associated with IGMP Snooping on the VLAN interface.

API Call

```
s_int32_t
igmp_if_snooping_unset (struct igmp_instance *igi,
                        struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

IGMP_ERR_SNOOP_DISABLE_FAILED when the snooping cannot be disabled because IGMP is enabled on the interface

igmp_if_snoop_fast_leave_set

This function enables fast-leave processing on the specified VLAN interface. Fast-leave processing is analogous to immediate-leave processing.

API Call

```
s_int32_t
igmp_if_snoop_fast_leave_set (struct igmp_instance *igi,
                              struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

igmp_if_snoop_fast_leave_unset

This function disables fast-leave processing on the specified VLAN interface.

API Call

```
s_int32_t
igmp_if_snoop_fast_leave_unset (struct igmp_instance *igi,
                                struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

igmp_if_snoop_mrouter_if_set

This function statically identifies a particular VLAN constituent interface as a multicast router (mrouter) interface for IGMP Snooping on the specified VLAN interface. This function may be invoked multiple times to configure multiple VLAN constituent interfaces as mrouter interfaces.

API Call

```
s_int32_t
igmp_if_snoop_mrouter_if_set (struct igmp_instance *igi,
                              struct interface *ifp,
                              u_int8_t *mrtr_ifname);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface
`mrtr_ifname`	Pointer to the interface-name string of the VLAN constituent interface to be identified as the mrouter interface

Output Parameters

None

Return Values

GMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

IGMP_ERR_CFG_WITH_MROUTE_PROXY when the interface is configured with mroute proxy; undo first

IGMP_ERR_OOM when a memory allocation error occurs

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

igmp_if_snoop_mrouter_if_unset

This function unsets the static configuration of a VLAN constituent interface as an mrouter interface on the specified VLAN interface.

API Call

```
s_int32_t
igmp_if_snoop_mrouter_if_unset (struct igmp_instance *igi,
                                struct interface *ifp,
                                u_int8_t *mrtr_ifname)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface
`mrtr_ifname`	Pointer to the interface-name string of the VLAN constituent interface to be identified as the mrouter interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

igmp_if_snoop_querier_set

This function enables IGMP Snooping Querier functionality on the specified VLAN interface.

API Call

```
s_int32_t
igmp_if_snoop_querier_set (struct igmp_instance *igi,
                           struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

igmp_if_snoop_querier_unset

This function disables IGMP Snooping Querier functionality on the specified VLAN interface.

API Call

```
s_int32_t
igmp_if_snoop_querier_unset (struct igmp_instance *igi,
                             struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

igmp_if_snoop_report_suppress_set

This function enables IGMP Snooping report-suppression on all constituent interfaces of the specified VLAN interface.

API Call

```
s_int32_t
igmp_if_snoop_report_suppress_set (struct igmp_instance *igi,
                                   struct interface *ifp);
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

igmp_if_snoop_report_suppress_unset

This function disables IGMP Snooping report-suppression on all constituent interfaces of the specified VLAN interface.

API Call

```
s_int32_t
igmp_if_snoop_report_suppress_unset (struct igmp_instance *igi,
                                     struct interface *ifp)
```

Input Parameters

`igi`	Pointer to the IGMP instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF when no such interface as the one specified is found

IGMP_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

IGMP_IF_ERR_ENABLE_FAILED when IGMP cannot be enabled because Snooping is disabled on the interface

CHAPTER 10 MRIBv4 IGMP MIB API

This chapter contains the Management Information Base (MIB) table API functions for the Multicast Routing Information Base IPv4 (MRIBv4) Internet Group Management Protocol (IGMP).

Include File

The `mrib4_igmp_snmp_api.h` file is the only one required to be included in *.c files outside of the `lib/mcast/mcast4/igmp/` directory where IGMP- related functionality is referenced. The function declaration for the following functions is made available by including `mrib4_igmp_snmp_api.h`.

Return Values

Each function can return an error code as shown in IGMP API Error Codes.

IGMP MIB Table

The IGMP interface table contains one row for each interface on which IGMP is enabled.

mrib4_igmp_snmp_api_if_querier_get

This function gets the querier address for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_querier_get (struct igmp_instance *igi,
                    struct interface **pifp,
                    u_int8_t **ret_var,
                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier address)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_querier_get_next

This function gets the querier address for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_querier_get_next (struct igmp_instance *igi,
                         struct interface **pifp,
                         u_int8_t **ret_var,
                         u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier address)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_query_interval_get

This function gets the query interval for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_query_interval_get (struct igmp_instance *igi,
                           struct interface **pifp,
                           u_int8_t **ret_var,
                           u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (query interval); 1 to 31744.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_query_interval_get_next

This function gets the query interval for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_query_interval_get_next (struct igmp_instance *igi,
                                  struct interface **pifp,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (query interval); 1 to 31744.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_status_get

This function gets the status of the entry corresponding to the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_status_get (struct igmp_instance *igi,
                     struct interface **pifp,
```

```
                    u_int8_t **ret_var,
                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (status):
	`IGMP_SNMP_ROW_STATUS_ACTIVE` 1
	`IGMP_SNMP_ROW_STATUS_NOTINSERVICE` 2
	`IGMP_SNMP_ROW_STATUS_NOTREADY` 3
	`IGMP_SNMP_ROW_STATUS_CREATEANDGO` 4
	`IGMP_SNMP_ROW_STATUS_CREATEANDWAIT` 5
	`IGMP_SNMP_ROW_STATUS_DESTROY` 6
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_status_get_next

This function gets the status of the entry corresponding to the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_status_get_next (struct igmp_instance *igi,
                        struct interface **pifp,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (status)
	`IGMP_SNMP_ROW_STATUS_ACTIVE` 1
	`IGMP_SNMP_ROW_STATUS_NOTINSERVICE` 2
	`IGMP_SNMP_ROW_STATUS_NOTREADY` 3

IGMP_SNMP_ROW_STATUS_CREATEANDGO 4

IGMP_SNMP_ROW_STATUS_CREATEANDWAIT 5

IGMP_SNMP_ROW_STATUS_DESTROY 6

`ret_var_len` Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_wrong_version_queries_get

This function gets the number of general queries received over the lifetime of the row entry, for which the IGMP version does not match the equivalent mgmdRouterInterfaceVersion.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_wrong_version_queries_get (struct igmp_instance *igi,
                                  struct interface **pifp,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`igi` IGMP instance

`pifp` Pointer to the interface pointer

Output Parameters

`ret_var` Return variable (version)

`ret_var_len` Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

GMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer 2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

igmp_if_wrong_version_queries_get_next

This function gets the number of general queries received over the lifetime of the row entry for which the IGMP version does not match the equivalent mgmdRouterInterfaceVersion.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_wrong_version_queries_get_next (struct igmp_instance *igi,
                                        struct interface **pifp,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (version)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

GMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer 2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_version_get

This function gets the version for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_version_get (struct igmp_instance *igi,
                    struct interface **pifp,
                    u_int8_t **ret_var,
                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (version). 1, 2, or 3.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_version_get_next

This function gets the version for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_version_get_next (struct igmp_instance *igi,
                          struct interface **pifp,
                          u_int8_t **ret_var,
                          u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (version). 1, 2, or 3,
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_query_response_interval_get

This function gets the query response interval for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_query_response_interval_get (struct igmp_instance *igi,
                                    struct interface **pifp,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (query response interval). 0 to 31744.

`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_query_response_interval_get_next

This function gets the query response interval for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_querier_get_next (struct igmp_instance *igi,
                          struct interface **pifp,
                          u_int8_t **ret_var,
                          u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (query response interval). 0 to 31744.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

igmp_if_querier_uptime_get

This function gets the querier uptime for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_querier_uptime_get (struct igmp_instance *igi,
                            struct interface **pifp,
                            u_int8_t **ret_var,
                            u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier uptime)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_querier_uptime_get_next

This function gets the querier uptime for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_querier_uptime_get_next (struct igmp_instance *igi,
                                struct interface **pifp,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier uptime)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_querier_expiry_time_get

This function gets the expiry time for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_querier_expiry_time_get (struct igmp_instance *igi,
                                    struct interface **pifp,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier expiry time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_querier_expiry_time_get_next

This function gets the expiry time for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_querier_expiry_time_get_next (struct igmp_instance *igi,
                                        struct interface **pifp,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier expiry time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_mroute_pxy_get

This function gets the proxy interface index for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_mroute_pxy_get (struct igmp_instance *igi,
                        struct interface **pifp,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (proxy interface index)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_mroute_pxy_get_next

This function gets the proxy interface index for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_mroute_pxy_get_next (struct igmp_instance *igi,
                            struct interface **pifp,
                            u_int8_t **ret_var,
                            u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance

`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (proxy interface index)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_robustness_var_get

This function gets the robustness variable for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_robustness_var_get (struct igmp_instance *igi,
                           struct interface **pifp,
                           u_int8_t **ret_var,
                           u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (robustness). 1 to 255.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_robustness_var_get_next

This function gets the robustness variable for the next interface.

Syntax

```
s_int32_t
```

```
mrib4_igmp_snmp_api_if_robustness_var_get_next (struct igmp_instance *igi,
                                  struct interface **pifp,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (robustness). 1 to 255.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_lmqi_get

This function gets the last member query interval for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_lmqi_get (struct igmp_instance *igi,
                 struct interface **pifp,
                 u_int8_t **ret_var,
                 u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (LMQI). 0 to 31744.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_lmqi_get_next

This function gets the last member query interval for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_lmqi_get_next (struct igmp_instance *igi,
                      struct interface **pifp,
                      u_int8_t **ret_var,
                      u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (LMQI) 0 to 31744
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_lmqc_get

This function gets the last-member query count (LMQC) for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_lmqc_get (struct igmp_instance *igi,
                  struct interface **pifp,
                  u_int8_t **ret_var,
                  u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (LMQC). 1 to 255.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_lmqc_get_next

This function gets the last-member query count for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_lmqc_get_next (struct igmp_instance *igi,
                      struct interface **pifp,
                      u_int8_t **ret_var,
                      u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (LMQC). 1 to 255.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_sqc_get

This function gets the start-up query count (SQC) for the specified interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_sqc_get (struct igmp_instance *igi,
                 struct interface **pifp,
                 u_int8_t **ret_var,
                 u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance

`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (SQC). 1 to 255.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_sqc_get_next

This function gets the start-up query count for the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_sqc_get_next (struct igmp_instance *igi,
                     struct interface **pifp,
                     u_int8_t **ret_var,
                     u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (SQC). 1 to 255.
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_sqi_get

This function gets the start-up query interval (SQI) for the specified interface.

Syntax

```
s_int32_t
```

```
mrib4_igmp_snmp_api_if_sqi_get (struct igmp_instance *igi,
                  struct interface **pifp,
                  u_int8_t **ret_var,
                  u_int32_t *ret_var_len)
```

Input Parameters

igi	IGMP instance
pifp	Pointer to the interface pointer

Output Parameters

ret_var Return variable (SQI). 1 to 31744.

ret_var_len Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_sqi_get_next

This function gets the start-up query interval for the next interface

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_sqi_get_next (struct igmp_instance *igi,
                      struct interface **pifp,
                      u_int8_t **ret_var,
                      u_int32_t *ret_var_len)
```

Input Parameters

igi	IGMP instance
pifp	Pointer to the interface pointer

Output Parameters

ret_var	Return variable (SQI). 1 to 31744.
ret_var_len	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

lmrib4_igmp_snmp_api_if_srclist_host_address_get

This function gets the host address for the specified interface, group address, and source address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_srclist_host_address_get (struct igmp_instance *igi,
                                    struct interface **pifp,
                                    struct pal_in4_addr group_addr,
                                    struct pal_in4_addr src_addr,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`group_addr`	Group address
`src_addr`	Source address

Output Parameters

`ret_var`	Return variable (host address)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_srclist_host_address_get_next

This function gets the host address for the next interface, group address, or source address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_srclist_host_address_get (struct igmp_instance *igi,
                                    struct interface **pifp,
                                    struct pal_in4_addr *group_addr,
                                    struct pal_in4_addr *src_addr,
                                    struct igmp_snmp_rtr_src_list_index *index,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`group_addr`	Group address
`src_addr`	Source address
`index`	

Output Parameters

`ret_var`	Return variable (host address)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_srclist_expiry_time_get

This function gets the time left prior to the expiration of the source list entry.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_srclist_expiry_time_get (struct igmp_instance *igi,
                                struct interface **pifp,
                                struct pal_in4_addr *group_addr,
                                struct pal_in4_addr *src_addr,
                                struct igmp_snmp_rtr_src_list_index *index,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`group_addr`	Group address
`src_addr`	Source address
`index`	

Output Parameters

`ret_var`	Return variable (expiry time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_srclist_expiry_time_get_next

This function gets the time left prior to the expiration of the next source list entry.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_srclist_expiry_time_get_next (struct igmp_instance *igi,
                                     struct interface **pifp,
                                     struct pal_in4_addr *group_addr,
                                     struct pal_in4_addr *src_addr,
                                     struct igmp_snmp_rtr_src_list_index *index,
                                     u_int8_t **ret_var,
                                     u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`group_addr`	Group address
`src_addr`	Source address
`index`	

Output Parameters

`ret_var`	Return variable (expiry time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_joins_get

This function gets the number of times a group membership has been added on this interface.

Syntax

```
s_int32_t
```

```
mrib4_igmp_snmp_api_if_joins_get (struct igmp_instance *igi,
                  struct interface **pifp,
                  u_int8_t **ret_var,
                  u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (version)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_joins_get_next

This function gets the number of times a group membership has been added on the next interface.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_joins_get_next (struct igmp_instance *igi,
                       struct interface **pifp,
                       u_int8_t **ret_var,
                       u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (version)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_groups_get

This function gets the current number of entries for this interface in the router cache table.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_groups_get (struct igmp_instance *igi,
                    struct interface **pifp,
                    u_int8_t **ret_var,
                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

mrib4_igmp_snmp_api_if_groups_get_next

This function gets the number of current entries for the next interface in the router cache table.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_groups_get_next (struct igmp_instance *igi,
                        struct interface **pifp,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP Cache MIB Table

The IGMP Cache Table contains one row for each IP multicast group for which there are members on a particular interface.

mrib4_igmp_snmp_api_if_cache_last_reporter_get

This function gets the last reporter for the specified interface and group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_last_reporter_get (struct igmp_instance *igi,
                                  struct interface **pifp,
                                  struct igmp_snmp_rtr_cache_index *index,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (last reporter)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_cache_last_reporter_get_next

This function gets the last reporter for the next interface or group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_last_reporter_get_next (struct igmp_instance *igi,
                                     struct interface **pifp,
                                     struct igmp_snmp_rtr_cache_index *index,
                                     u_int8_t **ret_var,
                                     u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp P`	pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (last reporter)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

GMP_ERR_NO_SUCH_GROUP_REC if no such group record is found

mrib4_igmp_snmp_api_if_cache_uptime_get

This function gets the uptime for the specified interface and group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_uptime_get (struct igmp_instance *igi,
                         struct interface **pifp,
                         struct igmp_snmp_rtr_cache_index *index,
                         u_int8_t **ret_var,
                         u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (uptime)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_cache_uptime_get_next

This function gets the uptime for the next interface or group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_uptime_get_next (struct igmp_instance *igi,
                                struct interface **pifp,
                                struct igmp_snmp_rtr_cache_index *index,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (uptime)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

mrib4_igmp_snmp_api_if_cache_expiry_time_get

This function gets the expiration time for the specified interface and group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_expiry_time_get (struct igmp_instance *igi,
```

```
                                struct interface **pifp,
                                struct igmp_snmp_rtr_cache_index *index,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (expiry time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_cache_expiry_time_get_next

This function gets the expiration time for the next interface or group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_expiry_time_get_next (struct igmp_instance *igi,
                                    struct interface **pifp,
                                    struct igmp_snmp_rtr_cache_index *index,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (expiry time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_cache_exclmode_exp_timer_get

This function gets the exclude mode expiration time for the specified interface and group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_exclmode_exp_timer_get (struct igmp_instance *igi,
                                        struct interface **pifp,
                                        struct igmp_snmp_rtr_cache_index *index,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (exclude mode expiry time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

mrib4_igmp_snmp_api_if_cache_exclmode_exp_timer_get_next

This function gets the exclude mode expiration time for the next interface or group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_exclmode_exp_timer_get_next (struct igmp_instance *igi,
                                            struct interface **pifp,
                                            struct igmp_snmp_rtr_cache_index *index,
                                            u_int8_t **ret_var,
                                              u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (exclude mode expiry time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

mrib4_igmp_snmp_api_if_cache_ver1_host_timer_get

This function gets the time remaining until the local router assumes that there are no more version 1 members on the IP subnet attached to the specified interface and group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_ver1_host_timer_get (struct igmp_instance *igi,
                                  struct interface **pifp,
                                  struct igmp_snmp_rtr_cache_index *index,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (version 1 host time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_cache_ver1_host_timer_get_next

This function gets the time remaining until the local router assumes that there are no more version 1 members on the IP subnet attached to the next interface or group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_ver1_host_timer_get_next (struct igmp_instance *igi,
                                        struct interface **pifp,
                                        struct igmp_snmp_rtr_cache_index *index,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (version 1 host time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_NO-SUCH_GROUP_REC if no such group record is found

mrib4_igmp_snmp_api_if_cache_ver2_host_timer_get

This function gets the time remaining until the local router assumes that there are no more version 2 members on the IP subnet attached to the specified interface and group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_ver2_host_timer_get (struct igmp_instance *igi,
                                    struct interface **pifp,
                                    struct igmp_snmp_rtr_cache_index *index,
                                      u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (version 2 host time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_cache_ver2_host_timer_get_next

This function gets the time remaining until the local router assumes that there are no more version 2 members on the IP subnet attached to the next interface or group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_ver2_host_timer_get_next (struct igmp_instance *igi,
                                      struct interface **pifp,
                                      struct igmp_snmp_rtr_cache_index *index,
                                      u_int8_t **ret_var,
                                      u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer

`index`

Output Parameters

`ret_var`	Return variable (version 2 host time)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

mrib4_igmp_snmp_api_if_cache_src_filter_mode_get

This function gets the source filter mode for the specified interface and group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_src_filter_mode_get (struct igmp_instance *igi,
                                  struct interface **pifp,
                                  struct igmp_snmp_rtr_cache_index *index,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (mode)
`INCLUDE`	1
`EXCLUDE`	2
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_cache_src_filter_mode_get_next

This function gets the source filter mode for the next interface or group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_cache_src_filter_mode_get_next (struct igmp_instance *igi,
                                        struct interface **pifp,
                                        struct igmp_snmp_rtr_cache_index *index,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (mode)
`INCLUDE`	1
`EXCLUDE`	2
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_NO_SUCH_GROUP_REC when no such group record is found

mrib4_igmp_snmp_api_if_inv_cache_address_get

This function gets the group address for the specified interface and group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_inv_cache_address_get (struct igmp_instance *igi,
                                struct interface **pifp,
                                struct igmp_snmp_inv_rtr_index *index,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance

`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (group address)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP_ERR_NO_SUCH_IFF if there is no such interface as the one specified

IGMP_ERR_API_GET if the function fails for any other reason

mrib4_igmp_snmp_api_if_inv_cache_address_get_next

This function gets the group address for the next interface or group address.

Syntax

```
s_int32_t
mrib4_igmp_snmp_api_if_inv_cache_address_get_next (struct igmp_instance *igi,
                                    struct interface **pifp,
                                    struct igmp_snmp_inv_rtr_index *index,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`igi`	IGMP instance
`pifp`	Pointer to the interface pointer
`index`	

Output Parameters

`ret_var`	Return variable (group address)
`ret_var_len`	Return variable length

Return Values

IGMP_ERR_NONE when the function succeeds

IGMP_ERR_INVALID_VALUE when there is no IGMP instance, the specified interface is NULL or no supported values are given

IGMP_ERR_L2_PHYSICAL_IF if the specified interface is a Layer-2 interface

IGMP API Error Codes

This section contains

- An alphabetical list of IGMP API error codes with descriptions
- A utility function (mrib4_igmp_snmp_api_strerror) to get an error code description

Error Code	Description
IGMP_ERR_BUF_TOO_SHORT	Processing exceeded buffer space
IGMP_ERR_CFG_FOR_PROXY_SERVICE	Interface configured for proxy service; undo first
IGMP_ERR_CFG_WITH_MROUTE_PROXY	Interface configured with mroute proxy; undo first
IGMP_ERR_DOOM	IGMP is doomed
IGMP_ERR_GENERIC	Operation failed
IGMP_ERR_IGMP_ENALBED	IGMP is enabled on a VLAN interface
IGMP_ERR_IF_GREC_LIMIT_REACHED	Group record limit reached on interface
IGMP_ERR_INVALID_AF	Invalid address family
IGMP_ERR_INVALID_COMMAND	Invalid command
IGMP_ERR_INVALID_FLAG	Invalid flag
IGMP_ERR_INVALID_VALUE	Invalid value
IGMP_ERR_L2_PHYSICAL_IF	Command is invalid on VLAN constituent interface
IGMP_ERR_L2_SOCK_FAIL	Layer-2 socket initialization failed
IGMP_ERR_L3_NON_VLAN_IF	Command is valid only on VLAN interfaces
IGMP_ERR_L3_SOCK_FAIL	Layer-3 socket initialization failed
IGMP_ERR_MALFORMED_ARG	Malformed argument
IGMP_ERR_MALFORMED_MSG	Malformed message received
IGMP_ERR_NO_CONTEXT_INFO	Failed to get VR/VRF context information
IGMP_ERR_NO_SUCH_GROUP_REC	No such group record found
IGMP_ERR_NO_SUCH_IFF	No such interface configured
IGMP_ERR_NO_SUCH_SOURCE_REC	No such source record found
IGMP_ERR_NO_SUCH_SVC_REG	No such service registration found
IGMP_ERR_NO_SUCH_VALUE	No such configured value found
IGMP_ERR_NO_VALID_CONFIG	Invalid configuration for this operation
IGMP_ERR_NONE	Operation successful
IGMP_ERR_OOM	Out of memory
IGMP_ERR_QI_LE_QRI	Query interval should be greater than query-response interval
IGMP_ERR_QRI_GT_QI	Query-response interval should be less than query interval
IGMP_ERR_SNOOP_ENABLED	IGMP Snooping is enabled on VLAN interface
IGMP_ERR_SOCK_JOIN_FAIL	Failed to join all PIM routers in multicast group
IGMP_ERR_TEMP_INTERNAL	Temporary internal run-time failure

Error Code	Description
IGMP_ERR_UNINIT_WITHOUT_DEREG	Cannot uninitialize without deregistration
IGMP_ERR_UNKNOWN_MSG	Unknown message

mrib4_igmp_snmp_api_strerror

This utility function may be used to obtain a descriptive string for the above defined error-codes.

Syntax

```
u_int8_t *
mrib4_igmp_snmp_api_strerror (s_int32_t iret)
```

Input Parameter

`iret`	IGMP error code

Return Value

Pointer to a constant character string which describes the error code.

CHAPTER 11 MRIBv6 MLD Command API

This chapter describes the functions for the Multicast Routing Information Base IPv6 (MRIBv6) Multicast Listener Discovery (MLD) protocol.

Include File

To use the functions in this chapter, you must include `mribd/mrib6/mld/mrib6_mld_api.h`.

Global Configuration API

This section contains the functions that enable the MRIBv6 MLD global-level commands.

mrib6_mld_api_limit_set

This function sets the limit for group-record states across all interfaces in the specified MLD Instance. An exception access-list can be specified to exclude certain groups from being subject to this limit value.

Syntax

```
mrib6_mld_api_limit_set (struct mld_instance *mli,
                u_int32_t limit,
                u_int8_t *except_alist);
```

Input Parameters

`mli`	Pointer to the MLD instance
`limit`	Limit value
`except_alist`	Pointer to an access-list name

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_OOM when a memory allocation error occurs

mrib6_mld_api_limit_unset

This function unsets the limit and exception access-list for group-record states across the specified MLD instance.

Syntax

```
int
mrib6_mld_api_limit_unset (struct mld_instance *mli);
```

Input Parameters

`mli`	Pointer to the MLD instance

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

mrib6_mld_api_ssm_map_enable_set

This function enables MLD Source-Specific Multicast (SSM) mapping at the Instance level.

Syntax

```
int
mrib6_mld_api_ssm_map_enable_set (struct mld_instance *mli)
```

Input Parameters

`mli`	Pointer to the MLD instance

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

mrib6_mld_api_ssm_map_enable_unset

This function disables SSM mapping.

Syntax

```
int
mrib6_mld_api_ssm_map_enable_unset (struct mld_instance *mli);
```

Input Parameters

`mli`	Pointer to the MLD instance

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

mrib6_mld_api_ssm_map_static_set

This function sets an SSM map definition. The supplied source-address string will be used to produce the (G, S) SSM mapping for the group address(es) defined as the supplied access-list reference string. This function may be invoked multiple times to define multiple SSM mappings.

Syntax

```
int
mrib6_mld_api_ssm_map_static_set (struct mld_instance *mli,
                        u_int8_t *alist,
                        u_int8_t *msrc_arg)
```

Input Parameters

`mli`	Pointer to the MLD instance
`alist`	Pointer to the group(s) address access-list name
`msrc_arg`	Pointer to the source-address string

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_OOM when a memory allocation error occurs

mrib6_mld_api_ssm_map_static_unset

This function unsets the SSM map definition identified by the supplied access-list reference string and source-address string.

Syntax

```
int
mrib6_mld_api_ssm_map_static_unset (struct mld_instance *mli,
                          u_int8_t *alist,
                          u_int8_t *msrc_arg);
```

Input Parameters

`mli`	Pointer to the MLD instance
`alist`	Pointer to the group(s) address access-list name

`msrc_arg`	Pointer to the source-address string

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_VALUE when no such configured value is found

Interface-Level Configuration API

This section contains the functions that enable MRIBv6 MLD interface-level configuration commands.

mrib6_mld_api_if_set

This function enables MLD on an interface.

Syntax

```
int
mrib6_mld_api_if_set (struct mld_instance *mli, struct interface *ifp)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_GENERIC when the operation fails

RESULT_OK when the function succeeds

mrib6_mld_api_if_unset

This function disables MLD on an interface.

Syntax

```
int
mrib6_mld_api_if_unset (struct mld_instance *mli, struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_GENERIC when the operation fails

MLD_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

RESULT_OK when the function succeeds

mrib6_mld_api_if_access_list_set

This function sets the MLD access-list.

Syntax

```
int
mrib6_mld_api_if_access_list_set (struct mld_instance *mli,
                        struct interface *ifp,
                        u_int8_t *alist)
```

Input Parameters

`mli`	Pointer to the MLD instance

`ifp`	Pointer to the interface
`alist`	Pointer to the access-list name

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_access_list_unset

This function sets the MLD access-list.

Syntax

```
int
mrib6_mld_api_if_access_list_unset (struct mld_instance *mli,
                          struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_immediate_leave_set

This function sets the MLD immediate-leave access-list.

Syntax

```
int
mrib6_mld_api_if_immediate_leave_set (struct mld_instance *mli,
                              struct interface *ifp,
                              u_int8_t *alist);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`alist`	Pointer to the access-list name

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_immediate_leave_unset

This function unsets the MLD immediate-leave access-list.

Syntax

```
int
mrib6_mld_api_if_immediate_leave_unset (struct mld_instance *mli,
                                        struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_limit_set

This function sets the limit for group-record states on the specified interface and MLD Instance. An exception access-list can be specified to exclude certain groups from being subject to this limit value.

Syntax

```
int
mrib6_mld_api_if_limit_set (struct mld_instance *mli,
                  struct interface *ifp,
                  u_int32_t limit,
                  u_int8_t *except_alist);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`limit`	Limit value
`except_alist`	Pointer to an access-list name

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_limit_unset

This function unsets the limit and exception access-list for group-record states on the specified interface and MLD instance.

Syntax

```
int
mrib6_mld_api_if_limit_unset (struct mld_instance *mli,
                    struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_lmqc_set

This function sets the last-member query-count (LMQC) value.

Syntax

```
int
mrib6_mld_api_if_lmqc_set (struct mld_instance *mli,
                  struct interface *ifp,
                  u_int32_t lmqc);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`lmqc`	The LMQC value

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_lmqc_unset

This function sets the last-member query-interval (LMQI) value.

Syntax

```
int
mrib6_mld_api_if_lmqc_unset (struct mld_instance *mli,
                    struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_lmqi_set

This function sets the last-member query-interval (LMQI) value.

Syntax

```
int
mrib6_mld_api_if_lmqi_set (struct mld_instance *mli,
                  struct interface *ifp,
                  u_int32_t lmqi);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`lmqi`	The LMQI value

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_lmqi_unset

This function unsets the last-member query-interval to the default value.

Syntax

```
int
mrib6_mld_api_if_lmqi_unset (struct mld_instance *mli,
                    struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_mroute_pxy_set

This function sets up the proxy-service interface association for the specified interface

Syntax

```
int
mrib6_mld_api_if_mroute_pxy_set (struct mld_instance *mli,
                        struct interface *ifp,
                        u_int8_t *mrtr_pxy_ifname)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the downstream interface in the MLD Proxy configuration
`mrtr_pxy_ifname`	Pointer to the interface name of the upstream proxy-service interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_CFG_FOR_PROXY_SERVICE when the interface is configured for proxy service; undo first

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_mroute_pxy_unset

This function unsets the proxy-service association on the specified downstream interface.

Syntax

```
int
mrib6_mld_api_if_mroute_pxy_unset (struct mld_instance *mli,
                          struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the downstream interface in the MLD Proxy configuration

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_pxy_service_set

This function sets the specified interface for MLD proxy service.

Syntax

```
int
mrib6_mld_api_if_pxy_service_set (struct mld_instance *mli,
                                  struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface that provides upstream host-side MLD Proxy-service

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_CFG_WITH_MROUTE_PROXY when the interface is configured with mroute proxy; undo first

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_pxy_service_unset

This function unsets MLD Proxy service on the specified interface.

Syntax

```
int
mrib6_mld_api_if_pxy_service_unset (struct mld_instance *mli,
                          struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface that provides upstream host-side MLD Proxy-service

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_querier_timeout_set

This function sets the MLD other-querier timeout.

Syntax

```
int
mrib6_mld_api_if_querier_timeout_set (struct mld_instance *mli,
                              struct interface *ifp,
                              u_int16_t other_querier_interval)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`other_querier_interval`	The value for the MLD other-querier timeout

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_querier_timeout_unset

This function sets the MLD other-querier timeout.

Syntax

```
int
mrib6_mld_api_if_querier_timeout_unset (struct mld_instance *mli,
                                struct interface *ifp)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_query_interval_set

This function sets the MLD query interval value.

Syntax

```
int
mrib6_mld_api_if_query_interval_set (struct mld_instance *mli,
                          struct interface *ifp,
                          u_int32_t query_interval);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`query_interval`	The MLD query interval value

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_query_interval_unset

This function unsets the MLD query interval value.

Syntax

```
int
mrib6_mld_api_if_query_interval_unset (struct mld_instance *mli,
                                       struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_QI_LE_QRI when the query interval should be greater than the query-response interval

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_query_response_interval_set

This function sets the MLD query-response interval value.

Syntax

```
int
```

```
mrib6_mld_api_if_query_response_interval_set (struct mld_instance *mli,
                                      struct interface *ifp,
                                      u_int32_t response_interval);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`response_interval`	The value of the MLD query-response interval

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_query_response_interval_unset

This function unsets the MLD query-response interval value.

Syntax

```
int
mrib6_mld_api_if_query_response_interval_unset (struct mld_instance *mli,
                                        struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_robustness_var_set

This function sets the robustness variable.

Syntax

```
int
mrib6_mld_api_if_robustness_var_set (struct mld_instance *mli,
                            struct interface *ifp,
                            u_int32_t robustness_var);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`robustness_var`	The robustness variable value

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_QRI_GT_QI when the query-response interval should be less than the query interval

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_robustness_var_unset

This function sets the robustness variable.

Syntax

```
int
mrib6_mld_api_if_robustness_var_unset (struct mld_instance *mli,
                                       struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_version_set

This function sets the MLD version.

Syntax

```
int
```

```
mrib6_mld_api_if_version_set (struct mld_instance *mli,
                    struct interface *ifp,
                    u_int16_t version);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`version`	The MLD version number

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_version_unset

This function unsets the MLD version.

Syntax

```
int
mrib6_mld_api_if_version_unset (struct mld_instance *mli,
                     struct interface *ifp);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

RESULT_OK when the function succeeds

MLD_ERR_GENERIC when the operation fails

mrib6_mld_api_if_static_group_source_set

This function sets the MLD static group.

Syntax

```
int
mrib6_mld_api_if_static_group_source_set (struct mld_instance *mli,
                                          struct interface *ifp,
                                          struct pal_in6_addr *pgrp,
                                          struct pal_in6_addr *psrc,
                                          u_int8_t *ifname,
                                          bool_t is_ssm_mapped);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`pgrp`	Pointer to the specific MLD group address
`psrc`	Pointer to the specific MLD source address
`ifname`	The interface name
`is_ssm_mapped`	Whether the entry is SSM-mapped

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

mrib6_mld_api_if_static_group_source_unset

This function sets the MLD static group.

Syntax

```
mrib6_mld_api_if_static_group_source_unset (struct mld_instance *mli,
                                            struct interface *ifp,
                                            struct pal_in6_addr *pgrp,
                                            struct pal_in6_addr *psrc,
                                            u_int8_t *ifname,
                                            bool_t is_ssm_mapped);
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the interface
`pgrp`	Pointer to the specific MLD group address
`psrc`	Pointer to the specific MLD source address
`ifname`	The interface name
`is_ssm_mapped`	Whether the entry is SSM-mapped

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

mrib6_mld_api_static_group_source_flag_unset

This function unsets the MLD static (S, G) group.

Syntax

```
void
mrib6_mld_api_static_group_source_flag_unset (struct mld_group_rec *mgr);
```

Input Parameters

`mgr`	MLD multicast group record entry

Output Parameters

None

Return Values

None

Clear Configuration API

This section contains the function that enables the MRIBv6 MLD `clear` command.

mrib6_mld_api_clear

This function clears MLD state information.

Syntax

```
int
mrib6_mld_api_clear (struct mld_instance *mli,
           struct interface *ifp,
           struct pal_in6_addr *pgrp,
           struct pal_in6_addr *psrc)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the specific interface to clear MLD information. If NULL, MLD group and source records on all interfaces belonging to the MLD instance are cleared.
`pgrp`	Pointer to the specific MLD group address to be cleared. If NULL, all MLD group and source records are cleared.
`psrc`	Pointer to the specific MLD source address to be cleared. If NULL, all MLD source records are cleared. Cannot be non-NULL when ifp or pgrp are NULL.

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_DOOM when the MLD protocol fails

CHAPTER 12 L2 MLD Snooping Command API

This chapter describes the L2 Multicast Listener Discovery protocol (MLD) snooping functions.

Include File

The `mld.h` file is the only one required to be included in `*.c` files outside of the `lib/mld/` directory where MLD-related functionality is referenced. The function declaration for the following APIs is made available by including `mld.h`.

Instance-Level Configuration API

This section contains the functions that enable instance-level commands.

mld_snooping_set

This function enables MLD Snooping on all interfaces of this instance not explicitly (individually) disabled for MLD Snooping. MLD Snooping is globally enabled by default.

API Call

```
s_int32_t
mld_snooping_set (struct mld_instance *mli)
```

Input Parameters

`mli`	Pointer to the MLD instance

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

mld_snooping_unset

This function disables MLD Snooping on all interfaces of this instance not explicitly (individually) enabled for MLD Snooping.

API Call

```
s_int32_t
mld_snooping_unset (struct mld_instance *mli)
```

Input Parameters

`mli`	Pointer to the MLD instance

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

Interface-Level Configuration API

This section contains the functions that enable interface-level configuration commands.

mld_if_snooping_set

This function explicitly enables MLD Snooping on the specified VLAN interface.

API Call

```
s_int32_t
mld_if_snooping_set (struct mld_instance *mli,
                     struct interface *ifp)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

MLD_ERR_GENERIC when the operation fails

RESULT_OK when the function succeeds

mld_if_snooping_unset

This function explicitly disables MLD Snooping on the specified VLAN interface. It also unsets all configuration associated with MLD Snooping on the VLAN interface.

API Call

```
s_int32_t
mld_if_snooping_unset (struct mld_instance *mli,
                       struct interface *ifp)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

RESULT_OK when the function succeeds

mld_if_snoop_fast_leave_set

This function enables fast-leave processing on the specified VLAN interface. Fast-leave processing is analogous to immediate-leave processing.

API Call

```
s_int32_t
mld_if_snoop_fast_leave_set (struct mld_instance *mli,
```

```
                                struct interface *ifp)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

MLD_ERR_GENERIC when the operation fails

RESULT_OK when the function succeeds

mld_if_snoop_fast_leave_unset

This function disables fast-leave processing on the specified VLAN interface.

API Call

```
s_int32_t
mld_if_snoop_fast_leave_unset (struct mld_instance *mli,
                               struct interface *ifp)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

MLD_ERR_GENERIC when the operation fails

RESULT_OK when the function succeeds

mld_if_snoop_mrouter_if_set

This function statically identifies a particular VLAN constituent interface as a multicast router (mrouter) interface for MLD Snooping on the specified VLAN interface. This function may be invoked multiple times to configure multiple VLAN constituent interfaces as mrouter interfaces.

API Call

```
s_int32_t
mld_if_snoop_mrouter_if_set (struct mld_instance *mli,
                             struct interface *ifp,
                             u_int8_t *mrtr_ifname)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface
`mrtr_ifname`	Pointer to the interface-name string of the VLAN constituent interface to be identified as the mrouter interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

MLD_ERR_GENERIC when the operation fails

RESULT_OK when the function succeeds

mld_if_snoop_mrouter_if_unset

This function unsets the static configuration of a VLAN constituent interface as an mrouter interface on the specified VLAN interface.

API Call

```
s_int32_t
mld_if_snoop_mrouter_if_unset (struct mld_instance *mli,
                               struct interface *ifp,
                               u_int8_t *mrtr_ifname)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface
`mrtr_ifname`	Pointer to the interface-name string of the VLAN constituent interface to be identified as the mrouter interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

RESULT_OK when the function succeeds

mld_if_snoop_querier_set

This function enables MLD Snooping Querier functionality on the specified VLAN interface.

API Call

```
s_int32_t
mld_if_snoop_querier_set (struct mld_instance *mli,
                          struct interface *ifp)
                          u_int8_t *mrtr_ifname)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

MLD_ERR_GENERIC when the operation fails

RESULT_OK when the function succeeds

mld_if_snoop_querier_unset

This function disables MLD Snooping Querier functionality on the specified VLAN interface.

API Call

```
s_int32_t
mld_if_snoop_querier_unset (struct mld_instance *mli,
                            struct interface *ifp)
                            u_int8_t *mrtr_ifname)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

MLD_ERR_GENERIC when the operation fails

RESULT_OK when the function succeeds

mld_if_snoop_report_suppress_set

This function enables MLD Snooping report-suppression on all constituent interfaces of the specified VLAN interface.

API Call

```
s_int32_t
mld_if_snoop_report_suppress_set (struct mld_instance *mli,
                                  struct interface *ifp)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

RESULT_OK when the function succeeds

mld_if_snoop_report_suppress_unset

This function disables MLD Snooping report-suppression on all constituent interfaces of the specified VLAN interface.

API Call

```
s_int32_t
mld_if_snoop_report_suppress_unset (struct mld_instance *mli,
                                    struct interface *ifp)
```

Input Parameters

`mli`	Pointer to the MLD instance
`ifp`	Pointer to the VLAN interface

Output Parameters

None

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_L3_NON_VLAN_IF when the command is valid only on VLAN interfaces

MLD_ERR_MALFORMED_ARG when the argument is malformed

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

MLD_ERR_NO_SUCH_SOURCE_REC when no such source record is found

MLD_ERR_OOM when a memory allocation error occurs

MLD_ERR_DOOM when the MLD protocol fails

MLD_ERR_TEMP_INTERNAL when a temporary internal run-time failure occurs

MLD_ERR_GENERIC when the operation fails

RESULT_OK when the function succeeds

CHAPTER 13 MRIBv6 MLD MIB API

This chapter contains the Management Information Base (MIB) table functions for the Multicast Routing Information Base IPv6 (MRIBv6) Internet Group Management Protocol (IGMP).

The MLD Interface MIB is defined in the IETF draft-ietf-ipngwg-mld-mib-05 document.

Include File

The `mrib6_mld_snmp_api.h` file is the only one required to be included in *.c files outside of the `lib/mcast/mcast6/mld/` directory where MLD-related functionality is referenced. The function declaration for the following functions is made available by including `mrib6_mld_snmp_api.h`.

Return Values

Each of the functions can return an error code. Error codes and definitions are in MLD API Error Codes.

API

mrib6_mld_snmp_api_if_querier_get

This function gets the querier address for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_querier_get (struct mld_instance *mli,
                   struct interface **pifp,
                   u_int8_t **ret_var,
                   pal_size_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier address)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_querier_get_next

This function gets the querier address for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_querier_get_next (struct mld_instance *mli,
                        struct interface **pifp,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier address)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_query_interval_get

This function gets the query interval for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_query_interval_get (struct mld_instance *mli,
                          struct interface **pifp,
                          u_int8_t **ret_var,
                          u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp P`	pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (query interval). 1 to 31744.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_query_interval_get_next

This function gets the query interval for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_query_interval_get_next (struct mld_instance *mli,
                                struct interface **pifp,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (query interval). 1 to 31744.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_status_get

This function creates or deletes a row in the router interface table.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_status_get (struct mld_instance *mli,
                  struct interface **pifp,
                  u_int8_t **ret_var,
                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (status)
`MLD_SNMP_ROW_STATUS_ACTIVE`	1
`MLD_SNMP_ROW_STATUS_NOTINSERVICE`	2
`MLD_SNMP_ROW_STATUS_NOTREADY`	3

MLD_SNMP_ROW_STATUS_CREATEANDGO 4

MLD_SNMP_ROW_STATUS_CREATEANDWAIT 5

MLD_SNMP_ROW_STATUS_DESTROY 6

ret_var_len Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_status_get_next

This function gets the status of the entry corresponding to the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_status_get_next (struct mld_instance *mli,
                        struct interface **pifp,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

mli MLD instance

pifp Pointer to the interface pointer

Output Parameters

ret_var Return variable (status)

MLD_SNMP_ROW_STATUS_ACTIVE 1

MLD_SNMP_ROW_STATUS_NOTINSERVICE 2

MLD_SNMP_ROW_STATUS_NOTREADY 3

MLD_SNMP_ROW_STATUS_CREATEANDGO 4

MLD_SNMP_ROW_STATUS_CREATEANDWAIT 5

MLD_SNMP_ROW_STATUS_DESTROY 6

ret_var_len Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_wrong_version_queries_get

This function gets the number of general queries received over the lifetime of the row entry, which have an MLD version that does not match the equivalent mgmdRouterInterfaceVersion.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_wrong_version_queries_get (struct mld_instance *mli,
                                  struct interface **pifp,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable status
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_wrong_version_queries_get_next

This function gets the number of general queries received over the lifetime of the row entry, which have an MLD version that does not match the equivalent mgmdRouterInterfaceVersion.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_wrong_version_queries_get_next (struct mld_instance *mli,
                                      struct interface **pifp,
                                      u_int8_t **ret_var,
                                      u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable status
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_joins_get

This function gets the number of times a group membership has been added on this interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_joins_get (struct mld_instance *mli,
                  struct interface **pifp,
                  u_int8_t **ret_var,
                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var` Return variable status

`ret_var_len` Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_joins_get_next

This function gets the number of times a group membership has been added on the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_joins_get_next (struct mld_instance *mli,
                      struct interface **pifp,
                      u_int8_t **ret_var,
                      u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable status
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_groups_get

This function gets the number of times a group membership has been added for this interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_groups_get (struct mld_instance *mli,
                  struct interface **pifp,
                  u_int8_t **ret_var,
                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable status
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_groups_get_next

This function gets the number of times a group membership has been added for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_groups_get_next (struct mld_instance *mli,
                        struct interface **pifp,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable status
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_version_get

This function gets the MLD version for the interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_version_get (struct mld_instance *mli,
                    struct interface **pifp,
                    u_int8_t **ret_var,
                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (version). 1, 2, or 3.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_version_get_next

This function gets the MLD version for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_version_get_next (struct mld_instance *mli,
                        struct interface **pifp,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (version). 1, 2, or 3.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_query_response_interval_get

This function gets the query response interval for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_query_response_interval_get (struct mld_instance *mli,
                                    struct interface **pifp,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance

`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (query response interval). 0 to 31744.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_query_response_interval_get_next

This function gets the query response interval for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_querier_get_next (struct mld_instance *mli,
                        struct interface **pifp,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (query response interval). 0 to 31744.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_querier_uptime_get

This function gets the querier uptime for the specified interface.

Syntax

```
s_int32_t
```

```
mrib6_mld_snmp_api_if_querier_uptme_get (struct mld_instance *mli,
                          struct interface **pifp,
                          u_int8_t **ret_var,
                          u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier uptime)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_querier_uptime_get_next

This function gets the querier uptime for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_querier_uptime_get_next (struct mld_instance *mli,
                                struct interface **pifp,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier uptime)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_querier_expiry_time_get

This function gets the expiration time for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_querier_expiry_time_get (struct mld_instance *mli,
                                struct interface **pifp,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier expiry time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_querier_expiry_time_get_next

This function gets the expiration time for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_querier_expiry_time_get_next (struct mld_instance *mli,
                                    struct interface **pifp,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (querier expiry time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_mroute_pxy_get

This function gets the proxy interface index for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_mroute_pxy_get (struct mld_instance *mli,
                      struct interface **pifp,
                      u_int8_t **ret_var,
                      u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (proxy interface index)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_mroute_pxy_get_next

This function gets the proxy interface index for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_mroute_pxy_get_next (struct mld_instance *mli,
                           struct interface **pifp,
                           u_int8_t **ret_var,
                           u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance

`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (proxy interface index)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_robustness_var_get

This function gets the robustness variable for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_robustness_var_get (struct mld_instance *mli,
                          struct interface **pifp,
                          u_int8_t **ret_var,
                          u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (robustness). 1 to 255.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_robustness_var_get_next

This function gets the robustness variable for the next interface.

Syntax

```
s_int32_t
```

```
mrib6_mld_snmp_api_if_robustness_var_get_next (struct mld_instance *mli,
                                struct interface **pifp,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (robustness). 1 to 255.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_lmqi_get

This function gets the last member query interval for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_lmqi_get (struct mld_instance *mli,
                struct interface **pifp,
                u_int8_t **ret_var,
                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (LMQI) 0 to 31744.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_lmqi_get_next

This function gets the last member query interval for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_lmqi_get_next (struct mld_instance *mli,
                      struct interface **pifp,
                      u_int8_t **ret_var,
                      u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (LMQI) 0 to 31744.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_lmqc_get

This function gets the last-member query count (LMQC) for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_lmqc_get (struct mld_instance *mli,
                 struct interface **pifp,
                 u_int8_t **ret_var,
                 u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (LMQC) 1 to 255.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_lmqc_get_next

This function gets the last-member query count for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_lmqc_get_next (struct mld_instance *mli,
                     struct interface **pifp,
                     u_int8_t **ret_var,
                     u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (LMQC) 1 to 255.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_sqc_get

This function gets the start-up query count (SQC) for the specified interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_sqc_get (struct mld_instance *mli,
                struct interface **pifp,
                u_int8_t **ret_var,
                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance

`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (SQC) to 255.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_sqc_get_next

This function gets the start-up query count for the next interface.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_sqc_get_next (struct mld_instance *mli,
                    struct interface **pifp,
                    u_int8_t **ret_var,
                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (SQC) to 255.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_sqi_get

This function gets the start-up query interval (SQI) for the specified interface.

Syntax

```
s_int32_t
```

```
mrib6_mld_snmp_api_if_sqi_get (struct mld_instance *mli,
                struct interface **pifp,
                u_int8_t **ret_var,
                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (SQI) 1 to 31744.
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_sqi_get_next

This function gets the start-up query interval for the next interface

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_sqi_get_next (struct mld_instance *mli,
                    struct interface **pifp,
                    u_int8_t **ret_var,
                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

Output Parameters

`ret_var`	Return variable (SQI) 1 to 31744.
`ret_var_len`	Return variable length

Output Parameters

`ret_var R`eturn variable (SQI). 1 to 31744.

`ret_var_len R`eturn variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

mrib6_mld_snmp_api_if_cache_last_reporter_get

This function gets the last reporter for the specified interface and group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_last_reporter_get (struct mld_instance *mli,
                                  struct interface **pifp,
                                  struct pal_in6_addr *cache_addr,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (last reporter)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_last_reporter_get_next

This function gets the last reporter for the next interface or group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_last_reporter_get_next (struct mld_instance *mli,
                                       struct interface **pifp,
                                       struct pal_in6_addr *cache_addr,
                                       u_int8_t **ret_var,
                                       u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (last reporter)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

mrib6_mld_snmp_api_if_cache_uptime_get

This function gets the uptime for the specified interface and group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_uptime_get (struct mld_instance *mli,
                        struct interface **pifp,
                        struct pal_in6_addr *cache_addr,
                        u_int8_t **ret_var,
                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (uptime)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_uptime_get_next

This function gets the uptime for the next interface or group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_uptime_get_next (struct mld_instance *mli,
                                struct interface **pifp,
                                struct pal_in6_addr *cache_addr,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (uptime)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUCH_GROUP_REC when no such group record is found

mrib6_mld_snmp_api_if_cache_expiry_time_get

This function gets the expiration time for the specified interface and group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_expiry_time_get (struct mld_instance *mli,
                                struct interface **pifp,
                                struct pal_in6_addr *cache_addr,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (expiry time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_expiry_time_get_next

This function gets the expiry time for the next interface or group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_expiry_time_get_next (struct mld_instance *mli,
                                  struct interface **pifp,
                                  struct pal_in6_addr *cache_addr,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (expiry time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_NO_SUC_GROUP_REC if no such group record is found

mrib6_mld_snmp_api_if_cache_exclmode_exp_timer_get

This function gets the exclude mode expiration time for the specified interface and group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_exclmode_exp_timer_get (struct mld_instance *mli,
                                        struct interface **pifp,
                                        struct pal_in6_addr *cache_addr,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (exclude mode expiry time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_exclmode_exp_timer_get_next

This function gets the exclude mode expiration time for the next interface or group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_exclmode_exp_timer_get_next (struct mld_instance *mli,
                                            struct interface **pifp,
                                            struct pal_in6_addr *cache_addr,
                                            u_int8_t **ret_var,
                                            u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (exclude mode expiry time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_ver1_host_timer_get

This function gets the time remaining until the local router assumes that there are no more version 1 members on the IP subnet attached to the specified interface and group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_ver1_host_timer_get (struct mld_instance *mli,
                                  struct interface **pifp,
                                  struct pal_in6_addr *cache_addr,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (version 1 host time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_ver1_host_timer_get_next

This function gets the time remaining until the local router assumes that there are no more version 1 members on the IP subnet attached to the next interface or group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_ver2_host_timer_get_next (struct mld_instance *mli,
                                        struct interface **pifp,
                                        struct pal_in6_addr *cache_addr,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (version 1 host time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_ver2_host_timer_get

This function gets the time remaining until the local router assumes that there are no more version 2 members on the IP subnet attached to the specified interface and group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_ver2_host_timer_get (struct mld_instance *mli,
                                    struct interface **pifp,
                                    struct pal_in6_addr *cache_addr,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (version 2 host time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_ver2_host_timer_get_next

This function gets the time remaining until the local router assumes that there are no more version 2 members on the IP subnet attached to the next interface or group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_ver2_host_timer_get_next (struct mld_instance *mli,
                                        struct interface **pifp,
                                        struct pal_in6_addr *cache_addr,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (version 2 host time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_src_filter_mode_get

This function gets the source filter mode for the specified interface and group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_src_filter_mode_get (struct mld_instance *mli,
                                    struct interface **pifp,
                                    struct pal_in6_addr *cache_addr,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (mode)
`INCLUDE`	1
`EXCLUDE`	2
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_cache_src_filter_mode_get_next

This function gets the source filter mode for the next interface or group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_cache_src_filter_mode_get_next (struct mld_instance *mli,
                                        struct interface **pifp,
                                        struct pal_in6_addr *cache_addr,
                                        u_int8_t **ret_var,
                                        u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer

`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (mode)
`INCLUDE`	1
`EXCLUDE`	2
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_inv_cache_address_get

This function gets the group address for the specified interface and group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_inv_cache_address_get (struct mld_instance *mli,
                                struct interface **pifp,
                                struct pal_in6_addr *cache_addr,
                                u_int8_t **ret_var,
                                u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (group address)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_inv_cache_address_get_next

This function gets the group address for the next interface or group address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_inv_cache_address_get_next (struct mld_instance *mli,
                                  struct interface **pifp,
                                  struct pal_in6_addr *cache_addr,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`cache_addr`	Group address

Output Parameters

`ret_var`	Return variable (group address)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_srclist_host_address_get

This function gets the host address for the specified interface, group address, and source address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_srclist_host_address_get (struct mld_instance *mli,
                                  struct interface **pifp,
                                  struct pal_in6_addr group_addr,
                                  struct pal_in6_addr src_addr,
                                  u_int8_t **ret_var,
                                  u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`group_addr`	Group address

`src_addr`	Source address

Output Parameters

`ret_var`	Return variable (host address)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_srclist_host_address_get_next

This function gets the host address for the next interface, group address, or source address.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_srclist_host_address_get_next (struct mld_instance *mli,
                                    struct interface **pifp,
                                    struct pal_in6_addr group_addr,
                                    struct pal_in6_addr src_addr,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`group_addr`	Group address
`src_addr`	Source address

Output Parameters

`ret_var`	Return variable (host address)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_srclist_expiry_time_get

This function gets the time left prior to the expiry of the source list entry.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_srclist_expiry_time_get (struct mld_instance *mli,
                                    struct interface **pifp,
                                    struct pal_in6_addr group_addr,
                                    struct pal_in6_addr src_addr,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance
`pifp`	Pointer to the interface pointer
`group_addr`	Group address
`src_addr`	Source address

Output Parameters

`ret_var`	Return variable (expiry time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

mrib6_mld_snmp_api_if_srclist_expiry_time_get_next

This function gets the time left prior to the expiry of the next source list entry.

Syntax

```
s_int32_t
mrib6_mld_snmp_api_if_srclist_expiry_time_get_next (struct mld_instance *mli,
                                    struct interface **pifp,
                                    struct pal_in6_addr group_addr,
                                    struct pal_in6_addr src_addr,
                                    u_int8_t **ret_var,
                                    u_int32_t *ret_var_len)
```

Input Parameters

`mli`	MLD instance

`pifp`	Pointer to the interface pointer
`group_addr`	Group address
`src_addr`	Source address

Output Parameters

`ret_var`	Return variable (expiry time)
`ret_var_len`	Return variable length

Return Values

MLD_ERR_NONE when the function succeeds

MLD_ERR_INVALID_VALUE when there is no MLD instance, the specified interface is NULL or no supported values are given

MLD_ERR_L2_PHYSICAL_IF when the specified interface is a Layer-2 interface

MLD_ERR_NO_SUCH_IFF when no such interface as the one specified is found

MLD_ERR_API_GET if the function fails for any other reason

MLD API Error Codes

This table contains:

- An alphabetical list of MLD API error codes, with descriptions
- A utility function (mrib6_mld_snmp_api_strerror) to get a description of an error code.

Error Code	Description
MLD_ERR_BUF_TOO_SHORT	Processing exceeded buffer space
MLD_ERR_CFG_FOR_PROXY_SERVICE	Interface configured for proxy service; undo first
MLD_ERR_CFG_WITH_MROUTE_PROXY	Interface configured with mroute proxy; undo first
MLD_ERR_DOOM	MLD is doomed
MLD_ERR_GENERIC	Operation failed
MLD_ERR_IF_GREC_LIMIT_REACHED	Group record limit reached on interface
MLD_ERR_INVALID_AF	Invalid address family
MLD_ERR_INVALID_COMMAND	Invalid command
MLD_ERR_INVALID_FLAG	Invalid flag
MLD_ERR_INVALID_VALUE	Invalid value
MLD_ERR_L2_PHYSICAL_IF	Command is invalid on VLAN constituent interface
MLD_ERR_L2_SOCK_FAIL	Layer-2 socket initialization failed
MLD_ERR_L3_NON_VLAN_IF	Command is valid only on VLAN interfaces
MLD_ERR_L3_SOCK_FAIL	Layer-3 socket initialization failed
MLD_ERR_MALFORMED_ARG	Malformed argument
MLD_ERR_MALFORMED_MSG	Malformed message received
MLD_ERR_NO_CONTEXT_INFO	Failed to get VR/VRF context information
MLD_ERR_NO_SUCH_GROUP_REC	No such group record found
MLD_ERR_NO_SUCH_IFF	No such interface configured
MLD_ERR_NO_SUCH_SOURCE_REC	No such source record found
MLD_ERR_NO_SUCH_SVC_REG	No such service registration found
MLD_ERR_NO_SUCH_VALUE	No such configured value found
MLD_ERR_NO_VALID_CONFIG	Invalid configuration for this operation
MLD_ERR_NONE	Operation successful
MLD_ERR_OOM	Out of memory
MLD_ERR_QI_LE_QRI	Query interval should be greater than query-response interval
MLD_ERR_QRI_GT_QI	Query-response interval should be less than query interval
MLD_ERR_SOCK_JOIN_FAIL	Failed to join all PIM routers in multicast group
MLD_ERR_TEMP_INTERNAL	Temporary internal run-time failure

Error Code	Description
MLD_ERR_UNINIT_WITHOUT_DEREG	Cannot uninitialize without deregistration
MLD_ERR_UNKNOWN_MSG	Unknown message

mrib6_mld_snmp_api_strerror

This utility function gets a description string for an error code.

Syntax

```
u_int8_t *
mrib6_mld_snmp_api_strerror (s_int32_t iret)
```

Input Parameter

`iret`	MLD error code

Output Parameters

None

Return Value

Pointer to a constant character string which describes the error code

CHAPTER 14 Layer 2 Multicast Routing Information Base API

In ZebOS-XP, the L2MRIBd process maintains snooping-related functionality.

Overview

The L2MRIBd process primarily interacts with:

- NSM, for bridge, VLAN and port- related information and their association with each other
- IMI for snooping related CLI's
- MSTPd for spanning tree related information
- MRIBd to find out whether IGMP is enabled on any VLAN
- HAL for updating snooping entries in hardware

A common IGMP/MLD library for both IGMP/MLD snooping and IGMP/MLD multicast router have been created.To create the L2MRIBd process, either IGMP or MLD snooping is enabled in the configuration file. This configuration file enables the HAVE_L2MRIBD build flag.

How L2mribd Interacts with Protocols

Figure 14-1 shows how the logical architecture of L2MRIBd communicates with other protocols.

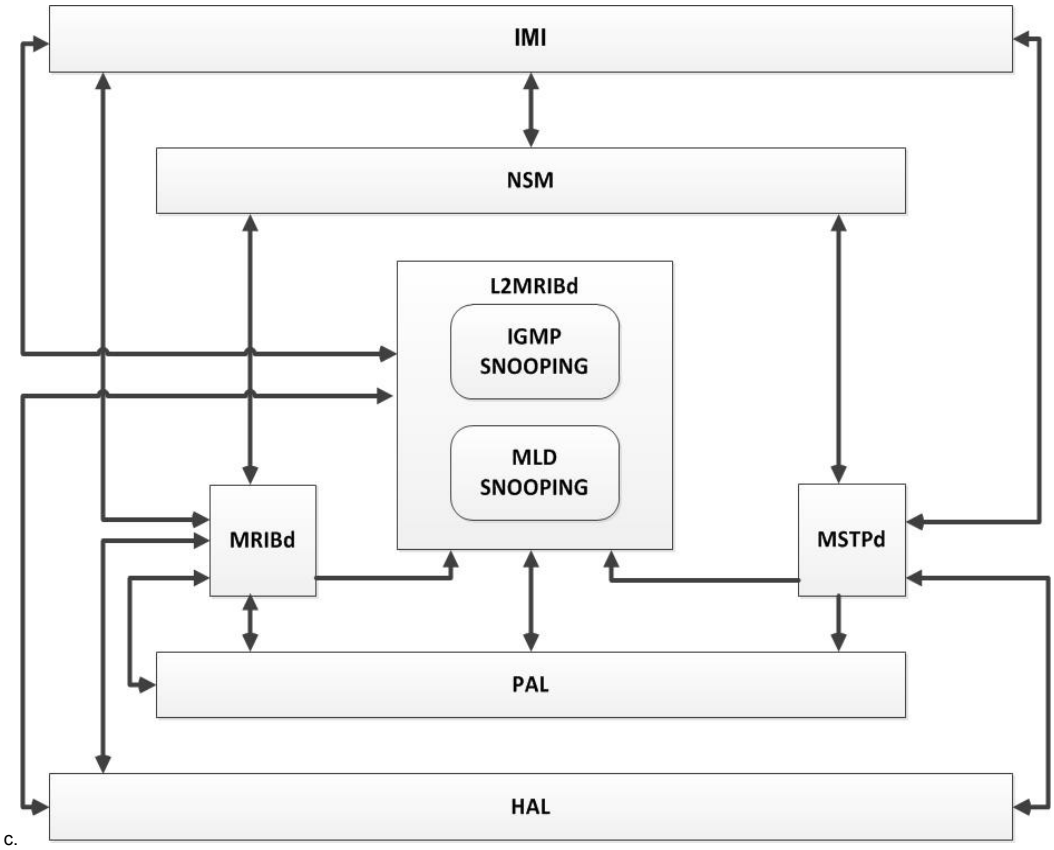

Figure 14-1: L2MRIB interacts with different protocols

- Initializes global libs and starts the protocol daemon.
- Starts NSM client to communicate with NSM.
- Creates the L2mcast, which holds the IGMP/MLD instances and associates input and output buffers.
- Starts the L2MRIB server to communicate with MSTPd.
- Starts the MRIB Client to communicate with MRIBd.
- Processes the IGMP/MLD join/leave/query messages.
- Updates the hardware with related snooping information.

Data Structures

This section describes the data structures for L2 MRIB.

l2mrib_master

This data structure in `l2mribd.h` holds an L2 related information.

Definition

```
struct l2mrib_master
{
  struct ipi_vr *vr;
```

```
  struct lib_globals *zg;

  struct l2mrib_mcast *l2mcast;

  struct list *mcast_bridge_list;

  struct list *bridge_config; /* struct br_config*/

  u_char config_flag;
#define CONFIG_IGMP_SNOOP_DISABLED            (1<<0)
#define CONFIG_MLD_SNOOP_DISABLED             (1<<1)

#ifdef HAVE_DISABLE_IGMP_SNOOP
#define CONFIG_IGMP_SNOOP_ENABLED             (1<<2)
#endif /* HAVE_DISABLE_IGMP_SNOOP */

#ifdef HAVE_DISABLE_MLD_SNOOP
#define CONFIG_MLD_SNOOP_ENABLED              (1<<3)
#endif /* HAVE_DISABLE_MLD_SNOOP */
};
```

l2mrib_mcast

This data structure in `l2mribd.h` maintains the details of IGMP/MLD snooping such as IGMP/MLD instances, input/output buffer, svc registration ID any many more.

Definition

```
struct l2mrib_mcast
{
  struct l2mrib_master *l2mm;

  /* Packet Input/Output buffer */
  struct stream *iobuf;

  /* Packet Output buffer */
  struct stream *obuf;

#ifdef HAVE_IGMP_SNOOP
  /* IGMP Instance */
  struct igmp_instance *igmp_inst;

  /* IGMP L2 Service Registration ID */
  void *igmp_svc_reg_id;
#endif

#ifdef HAVE_MLD_SNOOP
  /* MLD Instance */
  struct mld_instance *mld_inst;
```

```
  /* MLD L2 Service Registration ID */
  void *mld_svc_reg_id;
#endif

enum
  {
    L2MRIB_UNKNOWN_MCAST_FLOOD = 0,
    L2MRIB_UNKNOWN_MCAST_DISCARD = 1,
  }l2mrib_unknown_mcast;
```

l2mrib_bridge

This structure in `l2mribd.h` maintains all the bridge-related information, which is updated by NSM, from the messages.

Definition

```
struct l2mrib_bridge
{
  struct l2mrib_master *l2mm;

  u_int8_t      bridge_name[L2MRIB_BRIDGE_NAME_LEN+1];
  u_int8_t      bridge_type;

  u_int8_t      is_enabled;

  struct avl_tree *port_list;

  struct avl_tree *br_inst_list;

  struct avl_tree *vlan_table;

  struct avl_tree *snoop_entry;
  struct thread *t_snoop_entry_send;
};
```

l2mrib_if

This structure in `l2mribd.h` maintains all the interface-related information, which updated by NSM, from messages from NSM.

Definition

```
struct l2mrib_if
{
  struct l2mrib_bridge      *br;

  struct l2mrib_port        *l2port;

  struct l2mrib_vlan        *vlan;
```



```
#ifdef HAVE_IGMP_SNOOP
  struct ptree *igmpsnp_gmr_tib;
#endif

#ifdef HAVE_MLD_SNOOP
  struct ptree *mldsnp_gmr_tib;
#endif
```

Index

A

API error codes
  IGMP 259
  MLD 331
APIs
  IGMP CLI 173, 213
  IGMP MIB 225
  MLD CLI 261, 287
  Multicast Routing MIB 119
  NSM MLD Interface MIB 297
AVE_DISABLE_IGMP_SNOOP 335

B

bridge_config 335

C

CONFIG_IGMP_SNOOP_DISABLED 335
CONFIG_IGMP_SNOOP_ENABLED 335
CONFIG_MLD_SNOOP_DISABLED 335
CONFIG_MLD_SNOOP_ENABLED 335

G

GARP Multicast Registration Protocol 23
GMP L2 Service Registration ID 335
gmp_if_cache_src_filter_mode_get 256
gmp_if_cache_ver2_host_timer_get 255
GMRP command APIs
  gmrp_clear_all_statistics 30
  gmrp_clear_per_vlan_statistics 30
  gmrp_disable 27
  gmrp_disable_port 28
  gmrp_enable 27
  gmrp_enable_port 28
  gmrp_get_per_vlan_statistics_details 29
  gmrp_set_fwd_all 31
  gmrp_set_registration 29
  gmrp_set_timer 26
gmrp_clear_all_statistics 26, 30
gmrp_clear_per_vlan_statistics 30
gmrp_disable 27
gmrp_disable_port 28
gmrp_enable 27
gmrp_enable_port 28
gmrp_get_per_vlan_statistics_details 29
gmrp_set_fwd_all 31
gmrp_set_registration 29
gmrp_set_timer 26, 31

H

HAVE_DISABLE_MLD_SNOOP 335

I

IGMP API Error Codes 259
IGMP API error codes 259
IGMP cache MIB APIs
  igmp_if_cache_exclmode_exp_timer_get 252
  igmp_if_cache_exclmode_exp_timer_get_next 252
  igmp_if_cache_expiry_time_get 249
  igmp_if_cache_expiry_time_get_next 250
  igmp_if_cache_last_reporter_get 247
  igmp_if_cache_last_reporter_get_next 247
  igmp_if_cache_src_filter_mode_get 256
  igmp_if_cache_src_filter_mode_get_next 257
  igmp_if_cache_uptime_get 248
  igmp_if_cache_uptime_get_next 249
  igmp_if_cache_ver1_host_timer_get 253
  igmp_if_cache_ver1_host_timer_get_next 254
  igmp_if_cache_ver2_host_timer_get 255
  igmp_if_cache_ver2_host_timer_get_next 255
  igmp_if_inv_cache_address_get 257
  igmp_if_inv_cache_address_get_next 258
IGMP CLI APIs 173, 213
  igmp_if_snoop_fast_leave_set 218
  igmp_if_snoop_fast_leave_unset 218
  igmp_if_snoop_mrouter_if_set 219
  igmp_if_snoop_mrouter_if_unset 220
  igmp_if_snoop_querier_set 221
  igmp_if_snoop_querier_unset 221
  igmp_if_snoop_report_suppress_set 222
  igmp_if_snoop_report_suppress_unset 223
  igmp_if_snooping_set 216
  igmp_if_snooping_unset 216
  igmp_snooping_set 215
  igmp_snooping_unset 215
  mrib4_igmp_api_limit_set 173
IGMP Interface MIB API 225
IGMP Interface MIB APIs
  igmp_if_groups_get 246
  igmp_if_groups_get_next 246
  igmp_if_joins_get 244
  igmp_if_joins_get_next 245
  igmp_if_lmqc_get 238
  igmp_if_lmqc_get_next 239
  igmp_if_lmqi_get 237
  igmp_if_lmqi_get_next 238
  igmp_if_mroute_pxy_get 235
  igmp_if_mroute_pxy_get_next 235
  igmp_if_querier_expiry_time_get 234
  igmp_if_querier_expiry_time_get_next 234

igmp_if_querier_get 225
igmp_if_querier_get_next 226
igmp_if_querier_uptime_get 232
igmp_if_querier_uptime_get_next 233
igmp_if_query_interval_get 226
igmp_if_query_interval_get_next 227
igmp_if_query_response_interval_get 231
igmp_if_query_response_interval_get_next 232
igmp_if_robustness_var_get 236
igmp_if_robustness_var_get_next 236
igmp_if_sqc_get 239
igmp_if_sqc_get_next 240
igmp_if_sqi_get 240
igmp_if_sqi_get_next 241
igmp_if_srclist_expiry_time_get 243
igmp_if_srclist_expiry_time_get_next 244
igmp_if_srclist_host_address_get 242
igmp_if_srclist_host_address_get_next 242
igmp_if_status_get 227
igmp_if_status_get_next 228
igmp_if_version_get 230
igmp_if_version_get_next 231
igmp_if_wrong_version_queries_get 229
igmp_if_wrong_version_queries_get_next 229
IGMP MIB APIs 225
igmp_if_cache_exclmode_exp_timer_get 252
igmp_if_cache_exclmode_exp_timer_get_next 252
igmp_if_cache_expiry_time_get 249
igmp_if_cache_expiry_time_get_next 250
igmp_if_cache_last_reporter_get 247
igmp_if_cache_last_reporter_get_next 247
igmp_if_cache_src_filter_mode_get_next 257
igmp_if_cache_uptime_get_next 249
igmp_if_cache_ver1_host_timer_get 253
igmp_if_cache_ver1_host_timer_get_next 254
igmp_if_cache_ver2_host_timer_get_next 255
igmp_if_groups_get 246
igmp_if_groups_get_next 246
igmp_if_inv_cache_address_get 257
igmp_if_inv_cache_address_get_next 258
igmp_if_joins_get 244
igmp_if_joins_get_next 245
igmp_if_lmqc_get 238
igmp_if_lmqc_get_next 239
igmp_if_lmqi_get 237
igmp_if_lmqi_get_next 238
igmp_if_mroute_pxy_get 235
igmp_if_mroute_pxy_get_next 235
igmp_if_querier_expiry_time_get 234
igmp_if_querier_expiry_time_get_next 234
igmp_if_querier_get 225
igmp_if_querier_get_next 226
igmp_if_querier_uptime_get 232
igmp_if_querier_uptime_get_next 233
igmp_if_query_interval_get 226
igmp_if_query_interval_get_next 227
igmp_if_query_response_interval_get 231
igmp_if_query_response_interval_get_next 232
igmp_if_robustness_var_get 236
igmp_if_robustness_var_get_next 236
igmp_if_snoop_fast_leave_set 218
igmp_if_snoop_fast_leave_unset 218
igmp_if_snoop_mrouter_if_set 219
igmp_if_snoop_mrouter_if_unset 220
igmp_if_snoop_querier_set 221
igmp_if_snoop_querier_unset 221
igmp_if_snoop_report_suppress_unset 223
igmp_if_snoop_report_supress_set 222
igmp_if_snooping_unset 216
igmp_if_sqc_get 239
igmp_if_sqc_get_next 240
igmp_if_sqi_get 240
igmp_if_sqi_get_next 241
igmp_if_srclist_expiry_time_get 243
igmp_if_srclist_expiry_time_get_next 244
igmp_if_srclist_host_address_get 242
igmp_if_srclist_host_address_get_next 242
igmp_if_status_get 227
igmp_if_status_get_next 228
igmp_if_version_get 230
igmp_if_version_get_next 231
igmp_if_wrong_version_queries_get 229
igmp_if_wrong_version_queries_get_next 229
igmp_snooping_set 215
igmp_snooping_unset 215
iigmp_if_cache_uptime_get 248
imgp_if_snooping_set 216
ipi_vr 334

L

l2mrib_bridge 336
l2mrib_if 336
l2mrib_master 334
l2mrib_mcast 335
l2mrib.h 334
Layer 2 Multicast Routing Information Base 333

M

mcast_bridge_list 335
message exchange
  multicast protocol and IGMP 19
MLD API error codes 331
MLD CLI APIs 261, 287
  mld_if_set 264
  mld_if_snoop_fast_leave_set 289
  mld_if_snoop_fast_leave_unset 290
  mld_if_snoop_mrouter_if_set 291
  mld_if_snoop_mrouter_if_unset 292
  mld_if_snoop_querier_set 293
  mld_if_snoop_querier_unset 293
  mld_if_snoop_report_suppress_set 294
  mld_if_snoop_report_suppress_unset 295
  mld_if_snooping_set 288
  mld_if_snooping_unset 289
  mld_snooping_set 287
  mld_snooping_unset 287

MLD MIB APIs
mld_if_cache_exclmode_exp_timer_get 321
mld_if_cache_exclmode_exp_timer_get_next 321
mld_if_cache_expiry_time_get 319
mld_if_cache_expiry_time_get_next 320
mld_if_cache_last_reporter_get 317
mld_if_cache_last_reporter_get_next 317
mld_if_cache_src_filter_mode_get 325
mld_if_cache_src_filter_mode_get_next 325
mld_if_cache_uptime_get 318
mld_if_cache_uptime_get_next 319
mld_if_cache_ver1_host_timer_get 322
mld_if_cache_ver1_host_timer_get_next 323
mld_if_cache_ver2_host_timer_get 323
mld_if_cache_ver2_host_timer_get_next 324
mld_if_groups_get 304
mld_if_groups_get_next 305
mld_if_inv_cache_address_get 326
mld_if_inv_cache_address_get_next 327
mld_if_joins_get 303
mld_if_joins_get_next 303
mld_if_lmqc_get 313
mld_if_lmqc_get_next 314
mld_if_lmqi_get 312
mld_if_lmqi_get_next 313
mld_if_mroute_pxy_get 310
mld_if_mroute_pxy_get_next 310
mld_if_querier_expiry_time_get 309
mld_if_querier_expiry_time_get_next 309
mld_if_querier_get 297
mld_if_querier_get_next 298
mld_if_querier_uptime_get 307
mld_if_querier_uptime_get_next 308
mld_if_query_interval_get 298
mld_if_query_interval_get_next 300
mld_if_query_response_interval_get_next 307
mld_if_query_response_interval_getmld_if_query_response_interval_get 306
mld_if_robustness_var_get 311
mld_if_robustness_var_get_next 311
mld_if_sqc_get 314
mld_if_sqc_get_next 315
mld_if_sqi_get 315
mld_if_sqi_get_next 316
mld_if_srclist_expiry_time_get 329
mld_if_srclist_expiry_time_get_next 329
mld_if_srclist_host_address_get 327
mld_if_srclist_host_address_get_next 328
mld_if_status_get 300
mld_if_status_get_next 301
mld_if_version_get 305
mld_if_version_get_next 306
mld_if_wrong_version_queries_get 302
mld_if_wrong_version_queries_get_next 302
mld_strerror 332
mld_if_cache_exclmode_exp_timer_get 321
mld_if_cache_exclmode_exp_timer_get_next 321
mld_if_cache_expiry_time_get 319
mld_if_cache_expiry_time_get_next 320
mld_if_cache_last_reporter_get 317
mld_if_cache_last_reporter_get_next 317
mld_if_cache_src_filter_mode_get 325
mld_if_cache_src_filter_mode_get_next 325
mld_if_cache_uptime_get 318
mld_if_cache_uptime_get_next 319
mld_if_cache_ver1_host_timer_get 322
mld_if_cache_ver1_host_timer_get_next 323
mld_if_cache_ver2_host_timer_get 323
mld_if_cache_ver2_host_timer_get_next 324
mld_if_groups_get 304
mld_if_groups_get_next 305
mld_if_inv_cache_address_get 326
mld_if_inv_cache_address_get_next 327
mld_if_joins_get 303
mld_if_joins_get_next 303
mld_if_lmqc_get 313
mld_if_lmqc_get_next 314
mld_if_lmqi_get 312
mld_if_lmqi_get_next 313
mld_if_mroute_pxy_get 310
mld_if_mroute_pxy_get_next 310
mld_if_querier_expiry_time_get 309
mld_if_querier_expiry_time_get_next 309
mld_if_querier_get 297
mld_if_querier_get_next 298
mld_if_querier_uptime_get 307
mld_if_querier_uptime_get_next 308
mld_if_query_interval_get 298
mld_if_query_interval_get_next 300
mld_if_query_response_interval_get_next 307
mld_if_robustness_var_get 311
mld_if_robustness_var_get_next 311
mld_if_set 264
mld_if_snoop_fast_leave_set 289
mld_if_snoop_fast_leave_unset 290
mld_if_snoop_mrouter_if_set 291
mld_if_snoop_mrouter_if_unset 292
mld_if_snoop_querier_set 293
mld_if_snoop_querier_unset 293
mld_if_snoop_report_suppress_set 294
mld_if_snoop_report_suppress_unset 295
mld_if_snooping_set 288
mld_if_snooping_unset 289
mld_if_sqc_get 314
mld_if_sqc_get_next 315
mld_if_sqi_get 315
mld_if_sqi_get_next 316
mld_if_srclist_expiry_time_get 329
mld_if_srclist_expiry_time_get_next 329
mld_if_srclist_host_address_get 327
mld_if_srclist_host_address_get_next 328
mld_if_status_get 300
mld_if_status_get_next 301
mld_if_version_get 305
mld_if_version_get_next 306
mld_if_wrong_version_queries_get 302
mld_if_wrong_version_queries_get_next 302
mld_snooping_set 287

mld_snooping_unset 287
mld_strerror 332
MMRP Command APIs
  gmrp_clear_all_statistics 26
  gmrp_set_timer 31
  mmrp_disable_instance
    mmrp_disable_instance 31
  mmrp_enable_instance
    mmrp_enable_instance 32
MRIB 15
MRIB services 17
mrib4_igmp_api_limit_set 173
multicast architecture overview 15
Multicast Modules
  GARP Multicast Registration Protocol 23
multicast protocol IGMP message exchange 19
multicast protocol integration 15, 33
multicast route 213, 333
Multicast Routing MIB APIs 119

N

NSM MLD Interface MIB APIs 297

P

prefix_ipv4 334
prefix_ipv6 335

R

rib_master 336

S

snooping 213, 333
Supported Tables
  IP Multicast Route Table 119
Supported TablesIP Multicast Routing Interface Table 119
Supported TablesIP Multicast Routing Next Hop Table 119

V

VR data structures 33

Z

ZebOS common data structures 33